Typedef

```
#include <stdio.h>


typedef int Integer;


int main(){
   Integer a =20;
   printf("a = %d\n",a);


   return 0;
}



#include <stdio.h>


typedef struct Employee{


   int Employee_id;


   int Department_Number;


   float loginTime;


   float logoutTime;


} emp;
```

```
int main(){

   emp emp1;

   emp1.Employee_id = 1234;

   emp1.Department_Number = 4;

   return 0;

}
```

```
# include <stdio.h>

typedef float *fp;

int main(){

   float pi = 3.14;
   fp ptr = &pi;
   printf("Pi = %f\n",*ptr);
```

```
    return 0;

}
```

```
# include <stdio.h>

typedef float *fp;

int main(){

    float pi = 3.14;
    fp ptr = &pi;
    printf("Pi = %f\n",*ptr);

    return 0;

}
```

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>

#define MAX_ITEMS 100

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

```
} Item;

void addItem(Item inventory[], int *count);
void updateItem(Item *item);
void displayInventory(Item inventory[], int count);

static int totalItemsAdded = 0;

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

  do {
    printf("\nInventory Management System\n");
    printf("1. Add New Item\n");
    printf("2. Update Item Details\n");
    printf("3. Display Inventory\n");
    printf("4. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    switch (choice) {
      case 1:
        if (count < MAX_ITEMS) {
          addItem(inventory, &count);
        } else {
          printf("Inventory is full!\n");
```

```
    }
    break;

case 2:
    if (count > 0) {
        int id, found = 0;
        printf("Enter Item ID to update: ");
        scanf("%d", &id);

        for (int i = 0; i < count; i++) {
            if (inventory[i].itemID == id) {
                updateItem(&inventory[i]);
                found = 1;
                break;
            }
        }

        if (!found) {
            printf("Item with ID %d not found.\n", id);
        }
    } else {
        printf("No items in the inventory to update.\n");
    }
    break;

case 3:
    displayInventory(inventory, count);
    break;
```

```
            case 4:
                printf("Exiting the system.\n");
                break;

            default:
                printf("Invalid choice! Please try again.\n");
        }
    } while (choice != 4);

    return 0;
}

void addItem(Item inventory[], int *count) {
    printf("Enter Item ID: ");
    scanf("%d", &inventory[*count].itemID);
    printf("Enter Item Name: ");
    scanf("%s", inventory[*count].itemName);
    printf("Enter Price: ");
    scanf("%f", &inventory[*count].price);
    printf("Enter Quantity: ");
    scanf("%d", &inventory[*count].quantity);

    (*count)++;
    totalItemsAdded++;

    printf("Item added successfully!\n");
    printf("Total items added so far: %d\n", totalItemsAdded);
```

```
}

void updateItem(Item *item) {
   printf("Updating details for Item ID %d\n", item->itemID);
   printf("Enter New Item Name: ");
   scanf("%s", item->itemName);
   printf("Enter New Price: ");
   scanf("%f", &item->price);
   printf("Enter New Quantity: ");
   scanf("%d", &item->quantity);

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

void displayInventory(Item inventory[], int count) {
   if (count == 0) {
      printf("Inventory is empty!\n");
      return;
   }

   printf("\nCurrent Inventory:\n");
   printf("%-10s %-20s %-10s %-10s\n", "Item ID", "Item Name", "Price", "Quantity");
   for (int i = 0; i < count; i++) {
      printf("%-10d %-20s %-10.2f %-10d\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>

#define MAX_ITEMS 10
#define MAX_ORDERS 5
#define DISCOUNT_RATE 0.10 // 10% discount

typedef struct {
   int orderID;
   char customerName[50];
   float items[MAX_ITEMS];
   int itemCount;
   float totalCost;
} Order;

float calculateTotalCost(Order order);
```

```
void applyDiscount(Order *order);
void processOrders(Order orders[], int orderCount);

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

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

    if (orderCount > MAX_ORDERS) {
        printf("Exceeds maximum number of orders (%d).\n", MAX_ORDERS);
        return 1;
    }

    for (int i = 0; i < orderCount; i++) {
        printf("\nEnter details for Order %d:\n", i + 1);
        printf("Order ID: ");
        scanf("%d", &orders[i].orderID);

        printf("Customer Name: ");
        scanf("%s", orders[i].customerName);

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

        if (orders[i].itemCount > MAX_ITEMS) {
            printf("Exceeds maximum number of items (%d).\n", MAX_ITEMS);
```

```
            return 1;
        }

        for (int j = 0; j < orders[i].itemCount; j++) {
            printf("Enter cost of item %d: ", j + 1);
            scanf("%f", &orders[i].items[j]);
        }
    }

    // Process orders
    processOrders(orders, orderCount);

    return 0;
}

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

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

void processOrders(Order orders[], m int orderCount) {
```

```
  for (int i = 0; i < orderCount; i++) {
    printf("\nProcessing Order %d:\n", orders[i].orderID);

    // Calculate total cost before discount
    orders[i].totalCost = calculateTotalCost(orders[i]);
    printf("Total cost before discount: %.2f\n", orders[i].totalCost);

    // Apply discount
    applyDiscount(&orders[i]);
    printf("Total cost after %.0f%% discount: %.2f\n", DISCOUNT_RATE * 100,
orders[i].totalCost);
  }
}
```

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>
```

```
#define MAX_FEEDBACKS 100

typedef struct {
   int customerID;
   char feedbackText[100];
   int rating;
} Feedback;

void addFeedback(Feedback feedbacks[], int *count) {
   printf("Enter Customer ID: ");
   scanf("%d", &feedbacks[*count].customerID);
   printf("Enter Feedback Text: ");
   getchar();
   fgets(feedbacks[*count].feedbackText, sizeof(feedbacks[*count].feedbackText),
stdin);
   printf("Enter Rating (1-5): ");
   scanf("%d", &feedbacks[*count].rating);
   (*count)++;
}

void displayFeedbackSummary(Feedback feedbacks[], int count) {
   for (int i = 0; i < count; i++) {
      printf("\nCustomer ID: %d\n", feedbacks[i].customerID);
      printf("Feedback: %s", feedbacks[i].feedbackText);
      printf("Rating: %d - ", feedbacks[i].rating);
      switch (feedbacks[i].rating) {
         case 5: printf("Excellent\n"); break;
         case 4: printf("Good\n"); break;
```

```
            case 3: printf("Average\n"); break;
            case 2: printf("Poor\n"); break;
            case 1: printf("Very Poor\n"); break;
            default: printf("Invalid Rating\n");
        }
    }
}

int main() {
    Feedback feedbacks[MAX_FEEDBACKS];
    int count = 0;
    int choice;

    do {
        printf("\n1. Add Feedback\n2. Display Feedback Summary\n3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1: addFeedback(feedbacks, &count); break;
            case 2: displayFeedbackSummary(feedbacks, count); break;
            case 3: printf("Exiting...\n"); break;
            default: printf("Invalid choice, try again.\n");
        }
    } while (choice != 3);

    return 0;
}
```

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;

const float CHARGE_CREDIT_CARD = 2.5;  // Fixed transaction charge in percentage
const float CHARGE_DEBIT_CARD = 1.5;
const float CHARGE_UPI = 1.0;
const float CHARGE_NET_BANKING = 2.0;

void processPayment(Payment *payment) {
   if (strcmp(payment->method, "Credit Card") == 0) {
```

```
        payment->transactionCharge = (CHARGE_CREDIT_CARD / 100) * payment->amount;
    } else if (strcmp(payment->method, "Debit Card") == 0) {
        payment->transactionCharge = (CHARGE_DEBIT_CARD / 100) * payment->amount;
    } else if (strcmp(payment->method, "UPI") == 0) {
        payment->transactionCharge = (CHARGE_UPI / 100) * payment->amount;
    } else if (strcmp(payment->method, "Net Banking") == 0) {
        payment->transactionCharge = (CHARGE_NET_BANKING / 100) * payment->amount;
    } else {
        printf("Invalid payment method.\n");
        payment->transactionCharge = 0;
    }
}

void displayPaymentDetails(const Payment payment) {
    printf("\nPayment Details:\n");
    printf("Method: %s\n", payment.method);
    printf("Amount: %.2f\n", payment.amount);
    printf("Transaction Charge: %.2f\n", payment.transactionCharge);
    printf("Total Amount (including charges): %.2f\n", payment.amount + payment.transactionCharge);
}

int main() {
    Payment payment;
    int choice;

    printf("Select Payment Method:\n");
```

```
    printf("1. Credit Card\n2. Debit Card\n3. UPI\n4. Net Banking\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

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

    switch (choice) {
        case 1: strcpy(payment.method, "Credit Card"); break;
        case 2: strcpy(payment.method, "Debit Card"); break;
        case 3: strcpy(payment.method, "UPI"); break;
        case 4: strcpy(payment.method, "Net Banking"); break;
        default: printf("Invalid choice.\n"); return 1;
    }

    processPayment(&payment);
    displayPaymentDetails(payment);

    return 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>

#define MAX_CART_ITEMS 100

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

void addItem(CartItem cart[], int *count) {
   printf("Enter Item ID: ");
   scanf("%d", &cart[*count].itemID);
   printf("Enter Item Name: ");
   getchar();
   fgets(cart[*count].itemName, sizeof(cart[*count].itemName), stdin);
   cart[*count].itemName[strcspn(cart[*count].itemName, "\n")] = '\0';
   printf("Enter Price: ");
   scanf("%f", &cart[*count].price);
   printf("Enter Quantity: ");
```

```
    scanf("%d", &cart[*count].quantity);
    (*count)++;
}

void removeItem(CartItem cart[], int *count, int itemID) {
    for (int i = 0; i < *count; i++) {
        if (cart[i].itemID == itemID) {
            for (int j = i; j < *count - 1; j++) {
                cart[j] = cart[j + 1];
            }
            (*count)--;
            printf("Item removed successfully.\n");
            return;
        }
    }
    printf("Item not found in the cart.\n");
}

void displayCart(CartItem cart[], int count) {
    if (count == 0) {
        printf("The cart is empty.\n");
        return;
    }
    for (int i = 0; i < count; i++) {
        printf("\nItem ID: %d\n", cart[i].itemID);
        printf("Item Name: %s\n", cart[i].itemName);
        printf("Price: %.2f\n", cart[i].price);
        printf("Quantity: %d\n", cart[i].quantity);
```

```
    }
}

int main() {
    CartItem cart[MAX_CART_ITEMS];
    int count = 0, choice, itemID;

    do {
        printf("\n1. Add Item\n2. Remove Item\n3. Display Cart\n4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1: addItem(cart, &count); break;
            case 2:
                printf("Enter Item ID to remove: ");
                scanf("%d", &itemID);
                removeItem(cart, &count, itemID);
                break;
            case 3: displayCart(cart, count); break;
            case 4: printf("Exiting...\n"); break;
            default: printf("Invalid choice, try again.\n");
        }
    } while (choice != 4);

    return 0;
}
```

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>

#define MAX_PRODUCTS 100

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

void searchByName(Product products[], int count, char name[]) {
   for (int i = 0; i < count; i++) {
      if (strcmp(products[i].productName, name) == 0) {
         printf("\nProduct ID: %d\n", products[i].productID);
         printf("Product Name: %s\n", products[i].productName);
```

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

void updateProduct(Product products[], int count, int productID) {
    for (int i = 0; i < count; i++) {
        if (products[i].productID == productID) {
            printf("Enter New Product Name: ");
            getchar();
            fgets(products[i].productName, sizeof(products[i].productName), stdin);
            products[i].productName[strcspn(products[i].productName, "\n")] = '\0';
            printf("Enter New Category: ");
            fgets(products[i].category, sizeof(products[i].category), stdin);
            products[i].category[strcspn(products[i].category, "\n")] = '\0';
            printf("Enter New Price: ");
            scanf("%f", &products[i].price);
            printf("Product updated successfully.\n");
            return;
        }
    }
    printf("Product not found.\n");
}

int main() {
```

```
    Product products[MAX_PRODUCTS];
    int count = 0, choice, productID;
    char name[50];

    do {
        printf("\n1. Add Product\n2. Search Product by Name\n3. Update Product\n4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter Product ID: ");
                scanf("%d", &products[count].productID);
                printf("Enter Product Name: ");
                getchar();
                fgets(products[count].productName, sizeof(products[count].productName), stdin);
                products[count].productName[strcspn(products[count].productName, "\n")] = '\0';
                printf("Enter Category: ");
                fgets(products[count].category, sizeof(products[count].category), stdin);
                products[count].category[strcspn(products[count].category, "\n")] = '\0';
                printf("Enter Price: ");
                scanf("%f", &products[count].price);
                count++;
                break;
            case 2:
                printf("Enter Product Name to search: ");
                getchar();
```

```
            fgets(name, sizeof(name), stdin);
            name[strcspn(name, "\n")] = '\0';
            searchByName(products, count, name);
            break;
        case 3:
            printf("Enter Product ID to update: ");
            scanf("%d", &productID);
            updateProduct(products, count, productID);
            break;
        case 4: printf("Exiting...\n"); break;
        default: printf("Invalid choice, try again.\n");
      }
   } while (choice != 4);

   return 0;
}
```

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.

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

#define MAX_SALES 100

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

void add(Sale sales[], int *count);
void generateReport(Sale sales[], int count);

int main() {
   Sale sales[MAX_SALES];  // Fixed typo: Changed `Sales` to `Sale`
   int count = 0, choice;

   while (1) {
      printf("------- Sales Report Generator ---------\n");
      printf("1) Add Sale\n");
      printf("2) Generate Sales Report\n");
      printf("3) Exit\n");
      printf("Enter your choice: ");
      scanf("%d", &choice);

      switch (choice) {
```

```
        case 1:
            add(sales, &count);
            break;
        case 2:
            generateReport(sales, count);
            break;
        case 3:
            printf("Exiting.....\n");
            return 0;  // Exit the program
        default:
            printf("Invalid choice. Please try again.\n");
      }
   }

   return 0;
}

void add(Sale sales[], int *count) {
   printf("\nEnter Sale ID: ");
   scanf("%d", &sales[*count].saleID);
   printf("Enter the Product Category: ");
   getchar();  // Clear the newline character from the input buffer
   fgets(sales[*count].productCategory, sizeof(sales[*count].productCategory), stdin);
   sales[*count].productCategory[strcspn(sales[*count].productCategory, "\n")] =
'\0';  // Remove trailing newline
   printf("Enter Sale Amount: ");
   scanf("%f", &sales[*count].amount);
   printf("Enter Sale Date (dd/mm/yyyy): ");
```

```
    getchar();  // Clear the newline character again
    fgets(sales[*count].date, sizeof(sales[*count].date), stdin);
    sales[*count].date[strcspn(sales[*count].date, "\n")] = '\0';  // Remove trailing newline

    (*count)++;
}

void generateReport(Sale sales[], int count) {
    float electronicsTotal = 0, clothingTotal = 0, groceriesTotal = 0, othersTotal = 0;

    for (int i = 0; i < count; i++) {
        if (strcmp(sales[i].productCategory, "Electronics") == 0) {
            electronicsTotal += sales[i].amount;
        } else if (strcmp(sales[i].productCategory, "Clothing") == 0) {
            clothingTotal += 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("Clothing: %.2f\n", clothingTotal);
    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>

#define MAX_CUSTOMERS 10
#define REWARD_RATE 0.25

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

```
void addCustomer(Customer customer[],int *count);
void update(Customer *customer);
void display(Customer customer[],int count);

int main(){
   Customer customer[MAX_CUSTOMERS];
   int count =0,choice;
   while(1){
      printf("\n------- Customer Loyalty Program ---------\n");
      printf("1) Add Customer\n");
      printf("2) Display Customers with Reward Points\n");
      printf("3) Exit\n");
      printf("Enter your choice: ");
      scanf("%d", &choice);

      switch (choice){
         case 1:
            addCustomer(customer,&count);
            break;
         case 2:
            display(customer,count);
            break;
         case 3:
            printf("Exiting .....\n");
            return 0;
            break;
         default:
            printf("Invalid Choice \n");
```

```
        }

    }
    return 0;

}
void addCustomer(Customer customer[],int *count){
    printf("Enter customer ID ");
    scanf("%d",&customer[*count].customerID);

    printf("Enter the customer Name ");
    getchar();
    scanf("%s",&customer[*count].name);
    printf("Enter Total Purchases: ");
    scanf("%f", &customer[*count].totalPurchases);

    update(&customer[*count]);  // Update reward points based on total purchases

    (*count)++;
}
void update(Customer *customer){
    customer->rewardPoints = (int)(customer->totalPurchases *REWARD_RATE);
}
void display(Customer customer[],int count){
    if(count ==0){
        printf("\n No customer added yet\n");
        return;
```

```
    }
    printf("\n Customer Details: \n");
    for (int i = 0; i < count; i++) {
        printf("Customer ID: %d\n", customer[i].customerID);
        printf("Name: %s\n", customer[i].name);
        printf("Total Purchases: %.2f\n", customer[i].totalPurchases);
        printf("Reward Points: %d\n", customer[i].rewardPoints);
        printf("----------------------------\n");
    }

}
```

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>
```

```
#define MAX_ITEMS 10

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

void addItem(WarehouseItem items[], int *count);
void updateStock(WarehouseItem *item);
void checkReorderStatus(WarehouseItem items[], int count);

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

   while (1) {
      printf("\n------- Warehouse Management System ---------\n");
      printf("1) Add Warehouse Item\n");
      printf("2) Update Stock Levels\n");
      printf("3) Check Reorder Status\n");
      printf("4) Exit\n");
      printf("Enter your choice: ");
      scanf("%d", &choice);

      switch (choice) {
```

```
        case 1:
            addItem(items, &count);
            break;
        case 2:
            if (count == 0) {
                printf("No items in the warehouse. Add items first.\n");
            } else {
                for (int i = 0; i < count; i++) {
                    printf("\nUpdating stock for Item ID: %d (%s)\n", items[i].itemID,
items[i].itemName);
                    updateStock(&items[i]);
                }
            }
            break;
        case 3:
            if (count == 0) {
                printf("No items in the warehouse. Add items first.\n");
            } else {
                checkReorderStatus(items, count);
            }
            break;
        case 4:
            printf("Exiting...\n");
            return 0;
        default:
            printf("Invalid choice. Please try again.\n");
    }
}
```

```
    return 0;
}

void addItem(WarehouseItem items[], int *count) {
    printf("\nEnter Item ID: ");
    scanf("%d", &items[*count].itemID);
    printf("Enter Item Name: ");
    getchar();  // Clear the newline character
    fgets(items[*count].itemName, sizeof(items[*count].itemName), stdin);
    items[*count].itemName[strcspn(items[*count].itemName, "\n")] = '\0';  // Remove
trailing newline
    printf("Enter Current Stock: ");
    scanf("%d", &items[*count].currentStock);
    printf("Enter Reorder Level: ");
    scanf("%d", &items[*count].reorderLevel);

    (*count)++;
}

void updateStock(WarehouseItem *item) {
    int stockChange;
    printf("Enter stock adjustment (positive to add, negative to subtract): ");
    scanf("%d", &stockChange);
    item->currentStock += stockChange;

    if (item->currentStock < 0) {
        item->currentStock = 0;
```

```
    }

    printf("Updated stock for %s: %d\n", item->itemName, item->currentStock);
}

void checkReorderStatus(WarehouseItem items[], int count) {
    printf("\nReorder Status:\n");
    for (int i = 0; i < count; i++) {
        printf("Item ID: %d, Name: %s, Current Stock: %d, Reorder Level: %d\n",
               items[i].itemID, items[i].itemName, items[i].currentStock, items[i].reorderLevel);

        if (items[i].currentStock <= items[i].reorderLevel) {
            printf("-> Status: Reorder Required\n");
        } else {
            printf("-> Status: Sufficient Stock\n");
        }
    }
}
```

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>

#define MAX_CATEGORIES 5

typedef struct {
   char category[30];
   float discountPercentage;
   char validTill[15];
} Discount;

void updateDiscount(Discount *discount);
void displayDiscounts(Discount discounts[], int count);

int main() {
   Discount discounts[MAX_CATEGORIES] = {
      {"Electronics", 10.0, "31/12/2025"},
      {"Clothing", 15.0, "31/12/2025"},
      {"Groceries", 5.0, "31/12/2025"},
      {"Furniture", 12.0, "31/12/2025"},
      {"Others", 8.0, "31/12/2025"}
   };

   int choice;

   while (1) {
```

```
    printf("\n------- Discount Management System ---------\n");
    printf("1) Update Discount\n");
    printf("2) Display Discounts\n");
    printf("3) Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    switch (choice) {
      case 1:
        for (int i = 0; i < MAX_CATEGORIES; i++) {
          printf("\nUpdating discount for category: %s\n", discounts[i].category);
          updateDiscount(&discounts[i]);
        }
        break;
      case 2:
        displayDiscounts(discounts, MAX_CATEGORIES);
        break;
      case 3:
        printf("Exiting...\n");
        return 0;
      default:
        printf("Invalid choice. Please try again.\n");
    }
  }

  return 0;
}
```

```
void updateDiscount(Discount *discount) {
   printf("Current discount for %s: %.2f%% (Valid Till: %s)\n", discount->category,
discount->discountPercentage, discount->validTill);
   printf("Enter new discount percentage: ");
   scanf("%f", &discount->discountPercentage);
   printf("Enter new valid till date (dd/mm/yyyy): ");
   getchar();
   scanf("%s", discount->validTill);
   printf("Updated discount for %s: %.2f%% (Valid Till: %s)\n", discount->category,
discount->discountPercentage, discount->validTill);
}

void displayDiscounts(Discount discounts[], int count) {
   printf("\nDiscount Details:\n");
   for (int i = 0; i < count; i++) {
      printf("Category: %s\n", discounts[i].category);
      printf("Discount: %.2f%%\n", discounts[i].discountPercentage);
      printf("Valid Till: %s\n\n", discounts[i].validTill);
   }
}
```

```
#include <stdio.h>

union main
```

```
{
 int a;
 char b;
};

union add
{
  int c[10];
  char d;
};

int main(){
   union  add x2;
   union main x1;

   printf("Address of x1.a = %p \n",&x1.a);
    printf("Address of x1.b = %p \n",&x1.b);
}


#include<stdio.h>
union num{
   int a;
   char b;
};
```

```
int main(){
   union num x1;
   union num *ptr = &x1;
   ptr->a = 20;
   ptr->b ='A';
   printf("Adress of x1.a = %p \n",(*ptr).a);
   printf("Adress of x1.b = %p \n",(*ptr).b);
    printf("value of A = %d \n",(*ptr).a);
   printf("Value of B= %c \n",(*ptr).b);

   return 0;
}
```

```
#include<stdio.h>
typedef union num{
   int a;
   char b;
}n;
```

```
int main(){
   n x1;
   n *ptr = &x1;
   ptr->a = 20;
   ptr->b ='A';
   printf("Adress of x1.a = %p \n",(*ptr).a);
```

```
   printf("Adress of x1.b = %p \n",(*ptr).b);
    printf("value of A = %d \n",(*ptr).a);
   printf("Value of B= %c \n",(*ptr).b);

   return 0;
}
```

nested union

```
# include <stdio.h>

struct Student{
   char section;
   int rollNum;

   union {
      float mathMarks;
      float chemMarks;

      float TotalMarks;
   }marks;
};

int main(){
   struct Student s1;
   s1.section = 'A';
   s1.rollNum =123;
```

```
   s1.marks.chemMarks =85.5;
   s1.marks.mathMarks =75.4;
   s1.marks.TotalMarks = s1.marks.chemMarks + s1.marks.mathMarks;
   printf("Student 1 : section = %c    ROll number  = %d\n" ,s1.section,s1.rollNum);
   printf("Math Marks = %.2f    chem Marks = %.2f   Total marks =
%.2f",s1.marks.mathMarks,s1.marks.chemMarks,s1.marks.TotalMarks);
}
```

Union

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 a;
   float b;
   char c;
};
int main(){
   union data n;
   n.a = 20;
   printf("Value of a = %d\n",n.a);
   n.b = 3.14;
   printf("Value of b = %f\n",n.b);
   n.c ='C';
   printf("Value of c = %c\n",n.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>

union student {
    int rollNumber;
    char name[50];
};

int main(){
    union student stud;
    int choice;

    while(1){
        printf("\n-------Student Data with Union-------\n");
        printf("1) Enter Roll Number\n");
        printf("2) Enter Name\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice)
```

```
    {
    case 1:
      printf("Enter roll Number ");
      scanf("%d",&stud.rollNumber);
      printf("------Student Details----------\n");
      printf("Roll Number = %d ",stud.rollNumber);
      break;
    case 2:
      printf("Enter Name  ");
      getchar();
      scanf("%s",stud.name);
      printf("------Student Details----------\n");
      printf("Name = %s ",stud.name);
      break;


    default:
      printf("\nInvalid choice");
      return 0;
      break;
    }

  }
}
```

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 units{
   float kilometers;
   float miles;
};

int main(){
   union units u;
   int choice;

   printf("\n--------Union for Measurement Units ----------\n");
   printf("\n 1 ) Conversion from Km to miles\n");
   printf("\n 2 ) Conversion from miles to Km\n");

   printf("ENter choice ");
   scanf("%d",&choice);

   if(choice ==1){
      printf("Enter the Distance in Kilometers \n");
      scanf("%f",&u.kilometers);
      u.miles = u.kilometers * 0.621371;
      printf("Equivalent Distance in mile: %.2f",u.miles);
   }
   else if(choice ==2){
```

```
    printf("Enter the Distance in miles \n");
    scanf("%f",&u.miles);
    u.kilometers = u.miles/ 0.621371;
    printf("Equivalent Distance in kilometer: %.2f",u.kilometers);
  }else{
    printf("Invalid choice ");
  }


}
```

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 area {
  float radius;
  struct {
    float length;
    float width;
  } rectangle;
};

int main() {
```

```
    union area A;
    printf("-----------Union for Shape Dimensions----------------\n");
    printf("1 => Area of Circle \n");
    printf("2 => Area of Rectangle \n");

    int choice;
    printf("Enter your choice: ");
    scanf("%d", &choice);

    if (choice == 1) {
        printf("Enter the radius of the circle: ");
        scanf("%f", &A.radius);
        float circleArea = 3.14159 * A.radius * A.radius;
        printf("The area of the Circle is: %.2f\n", circleArea);
    } else if (choice == 2) {
        printf("Enter the length of the rectangle: ");
        scanf("%f", &A.rectangle.length);
        printf("Enter the width of the rectangle: ");
        scanf("%f", &A.rectangle.width);
        float rectangleArea = A.rectangle.length * A.rectangle.width;
        printf("The area of the Rectangle is: %.2f\n", rectangleArea);
    } else {
        printf("Invalid choice. Please enter 1 or 2.\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 Employee {
   int ID;
   float salary;
};

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


   printf("\n-------Union for Employee Data-------\n");
   printf("1) Employee ID\n");
   printf("2) Employee Salary\n");
   printf("Enter your choice: ");
   scanf("%d", &choice);


   if(choice ==1){
```

```
    printf("Enter ID ");
    scanf("%d",&stud.ID);
    printf("------Employee Details----------\n");
    printf("EMployee ID = %d ",stud.ID);
  }
  else if(choice ==2){
    printf("Enter Salary  ");
    scanf("%f",&stud.salary);
    printf("------EMployee Details----------\n");
    printf("salary  = %.2f",stud.salary);
  }


  else{
      printf("\nInvalid choice");
      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(){
   union Sensor S;

  S.temperature = 75.5;
   printf("Temperature: %.2f C\n", S.temperature);

   S.pressure = 123.3;
   printf("Pressure: %.2f Pa\n", S.pressure);

   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;
};

int main(){
   union Account A;
   int choice;
   printf("---------- Union for Bank Account Information----------\n");
   printf("1=> Display Account number \n");
   printf("2=> Display balance ");
   printf("Enter a Choice \n");
   scanf("%d",&choice);
   if(choice == 1){
      printf("Enter the account number ");
      scanf("%d",&A.accountNumber);
      printf("-----------------------------------------\n");
      printf("Your Account number is %d\n",A.accountNumber);
   }
   else if(choice ==2){
      printf("Enter your current balance \n");
      scanf("%f",&A.balance);
      printf("-----------------------------------------\n");
```

```
        printf("Your current balance is %.2f\n",A.balance);


    }
    else{
        printf("Invalid Choice \n");
    }



}
```

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 Vehicle {
    int regNum;
    float fuelCapacity;
};

int main(){
    union Vehicle A;
    int choice;
    printf("---------- Union for Vehicle Information----------\n");
```

```
    printf("1=> Display registration number \n");
    printf("2=> Display fuel Capacity ");
    printf("Enter a Choice \n");
    scanf("%d",&choice);
    if(choice == 1){
        printf("Enter the registration number ");
        scanf("%d",&A.regNum);
        printf("-----------------------------------------\n");
        printf("Your registration number is %d\n",A.regNum);
    }
    else if(choice ==2){
        printf("Enter your current fuel capacity \n");
        scanf("%f",&A.fuelCapacity);
        printf("-----------------------------------------\n");
        printf("Your current fuel capacity is %.2f\n",A.fuelCapacity);

    }
    else{
        printf("Invalid Choice \n");
    }


}
```

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.

```c
#include <stdio.h>

union ExamResults {
   int marks;
   char grade;
};

int main() {
   union ExamResults result;
   int choice;



   printf("---------- Union for Exam Results ----------\n");
   printf("1 => Display Marks\n");
   printf("2 => Display Grade\n");
   printf("Enter a choice: ");
   scanf("%d", &choice);

   if (choice == 1) {
      printf("Enter the marks: ");
      scanf("%d", &result.marks);
      printf("-------------------------------------------\n");
      printf("Student Marks: %d\n", result.marks);
   }
   else if (choice == 2) {
      printf("Enter the grade: ");
      scanf(" %c", &result.grade);
```

```
        printf("-----------------------------------------\n");
        printf("Student Grade: %c\n", result.grade);
    }
    else {
        printf("Invalid choice.\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;
};

int main() {
    union Currency currency;
    int choice;
    float conversionRate = 0.85;

    printf("---------- Union for Currency Conversion ----------\n");
```

```
    printf("1 => Convert USD to EUR\n");
    printf("2 => Convert EUR to USD\n");
    printf("Enter a choice: ");
    scanf("%d", &choice);

    if (choice == 1) {
        printf("Enter the value in USD: ");
        scanf("%f", &currency.usd);
        printf("-----------------------------------------\n");
        printf("Equivalent in EUR: %.2f\n", currency.usd * conversionRate);
    }
    else if (choice == 2) {
        printf("Enter the value in EUR: ");
        scanf("%f", &currency.eur);
        printf("-----------------------------------------\n");
        printf("Equivalent in USD: %.2f\n", currency.eur / conversionRate);
    }
    else {
        printf("Invalid choice.\n");
    }

    return 0;
}
```

Task 2:

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[50];
   int capacity;
   char status[50];
};

static int totalAircrafts = 0; // Static variable to track total aircrafts
```

```
void addAircraft(struct Aircraft *fleet, int *count);
void update(struct Aircraft *fleet, int count);
void display(struct Aircraft *fleet, int count);

int main() {
   struct Aircraft fleet[100];
   int count = 0;

   while (1) {
      printf("\n----- Aircraft Fleet Management -----\n");
      printf("1. Add new Aircraft\n");
      printf("2. Update Aircraft Status\n");
      printf("3. Display Fleet Information\n");
      printf("4. Exit\n");
      printf("Enter your choice: ");
      scanf("%d", &count);

      switch (count) {
         case 1:
            addAircraft(fleet, &totalAircrafts);
            break;
         case 2:
            update(fleet, totalAircrafts);
            break;
         case 3:
            display(fleet, totalAircrafts);
            break;
         case 4:
```

```
            printf("Exiting the system.\n");
            return 0;
        default:
            printf("Invalid choice, please try again.\n");
    }
  }
  return 0;
}

void addAircraft(struct Aircraft *fleet, int *count) {
  if (*count >= 100) {
    printf("Fleet has reached its maximum capacity.\n");
    return;
  }

  struct Aircraft new;
  printf("Enter Aircraft ID: ");
  scanf("%d", &new.aircraftID);

  getchar(); // Clear the newline character left by scanf

  printf("Enter Aircraft Model: ");
  scanf("%[^\n]%*c", new.model);  // Read the entire line including spaces until newline

  printf("Enter Aircraft Capacity: ");
  scanf("%d", &new.capacity);

  printf("Enter Aircraft Status (e.g., 'In Service', 'Out of Service'): ");
```

```
    scanf("%s", new.status);

    fleet[*count] = new;
    (*count)++;
    totalAircrafts++;
}

void update(struct Aircraft *fleet, int count) {
    int id;
    printf("Enter the Aircraft ID to update the status: ");
    scanf("%d", &id);

    int found = 0;
    for (int i = 0; i < count; i++) {
        if (fleet[i].aircraftID == id) {
            found = 1;
            printf("Enter new status for Aircraft ID %d: ", id);
            getchar();
            scanf("%s", fleet[i].status);
            break;
        }
    }

    if (!found) {
        printf("Aircraft ID not found.\n");
    }
}
```

```
void display(struct Aircraft *fleet, int count) {
   printf("\n----- Aircraft Fleet Information -----\n");
   for (int i = 0; i < count; i++) {
      printf("Aircraft ID: %d\n", fleet[i].aircraftID);
      printf("Model: %s\n", fleet[i].model);
      printf("Capacity: %d\n", fleet[i].capacity);
      printf("Status: %s\n", fleet[i].status);
      printf("---------------------------------------\n");
   }
}
```

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>

#define MAX_IMAGE_SIZE 100
#define MAX_TELEMETRY 10
```

```
struct Telemetry {
   float temperature;
   float velocity;
   float altitude;
};

union SatelliteData {
   char image[MAX_IMAGE_SIZE];
   struct Telemetry telemetry;
};

const float MAX_TEMP = 100.0f;
const float MAX_VELOCITY = 5000.0f;
const float MAX_ALTITUDE = 100000.0f;

void processImageData(const union SatelliteData* satelliteData);
void processTelemetryData(const union SatelliteData* satelliteData);

int main(){
   union SatelliteData satelliteData[MAX_TELEMETRY];
   int choice;
   int telemetryCount = 0;
   while (1) {
      printf("\n----- Satellite Data Processing -----\n");
      printf("1. Process Image Data\n");
      printf("2. Process Telemetry Data\n");
      printf("3. Exit\n");
```

```
        printf("Enter your choice: ");
        scanf("%d", &choice);
        getchar();  // Consume newline character after scanf

        switch (choice) {
            case 1: {
                if (telemetryCount < MAX_TELEMETRY) {
                    printf("Enter image data (string): ");
                    fgets(satelliteData[telemetryCount].image, MAX_IMAGE_SIZE, stdin);
                    satelliteData[telemetryCount].image[strcspn(satelliteData[telemetryCount].i
mage, "\n")] = '\0'; // Remove newline at the end
                    processImageData(&satelliteData[telemetryCount]);
                    telemetryCount++;
                } else {
                    printf("Max image data points reached.\n");
                }
                break;
            }
            case 2: {
                if (telemetryCount < MAX_TELEMETRY) {
                    printf("Enter temperature (in Celsius): ");
                    scanf("%f", &satelliteData[telemetryCount].telemetry.temperature);
                    printf("Enter velocity (in m/s): ");
                    scanf("%f", &satelliteData[telemetryCount].telemetry.velocity);
                    printf("Enter altitude (in meters): ");
                    scanf("%f", &satelliteData[telemetryCount].telemetry.altitude);
                    processTelemetryData(&satelliteData[telemetryCount]);
                    telemetryCount++;
```

```
            } else {
                printf("Max telemetry data points reached.\n");
            }
            break;
        }
        case 3:
            printf("Exiting the system.\n");
            return 0;
        default:
            printf("Invalid choice, please try again.\n");
    }
  }

  return 0;
}

void processImageData(const union SatelliteData* satelliteData) {
  printf("\nProcessing Image Data: %s\n", satelliteData->image);
}

void processTelemetryData(const union SatelliteData* satelliteData) {
  printf("\nProcessing Telemetry Data:\n");
  printf("Temperature: %.2f\n", satelliteData->telemetry.temperature);
  printf("Velocity: %.2f\n", satelliteData->telemetry.velocity);
  printf("Altitude: %.2f\n", satelliteData->telemetry.altitude);

  if (satelliteData->telemetry.temperature > MAX_TEMP) {
    printf("Warning: Temperature exceeds maximum limit.\n");
```

```
    }
    if (satelliteData->telemetry.velocity > MAX_VELOCITY) {
        printf("Warning: Velocity exceeds maximum limit.\n");
    }
    if (satelliteData->telemetry.altitude > MAX_ALTITUDE) {
        printf("Warning: Altitude exceeds maximum 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>

#define MAX_MISSIONS 10
#define MAX_CREW 5
#define MAX_CARGO 50
```

```
struct Crew {
   char name[50];
   int age;
   char role[50];
};

struct Cargo {
   char cargoName[50];
   float weight;
};

union Payload {
   struct Crew crew[MAX_CREW];
   struct Cargo cargo[MAX_CARGO];
};

struct Mission {
   int missionID;
   char name[50];
   int duration;
   union Payload payload;
   char missionType[20];
};

static int totalMissions = 0;

void addMission(struct Mission *missions, int *count);
void updateMission(struct Mission *missions, int count);
```

```
void displayMissionSummary(const struct Mission *missions, int count);
void manageCrewMission(struct Mission *mission);
void manageCargoMission(struct Mission *mission);

int main() {
    struct Mission missions[MAX_MISSIONS];
    int choice;
    int missionCount = 0;

    while (1) {
        printf("\n----- Mission Control System -----\n");
        printf("1. Add Mission\n");
        printf("2. Update Mission\n");
        printf("3. Display Mission Summaries\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        getchar();

        switch (choice) {
            case 1:
                addMission(missions, &missionCount);
                break;
            case 2:
                updateMission(missions, missionCount);
                break;
            case 3:
                displayMissionSummary(missions, missionCount);
```

```
            break;
        case 4:
            printf("Exiting the system.\n");
            return 0;
        default:
            printf("Invalid choice. Please try again.\n");
      }
   }

   return 0;
}

void addMission(struct Mission *missions, int *count) {
   if (*count >= MAX_MISSIONS) {
      printf("Max missions reached.\n");
      return;
   }

   struct Mission newMission;
   printf("Enter Mission ID: ");
   scanf("%d", &newMission.missionID);
   getchar();

   printf("Enter Mission Name: ");
   fgets(newMission.name, sizeof(newMission.name), stdin);
   newMission.name[strcspn(newMission.name, "\n")] = '\0';

   printf("Enter Mission Duration (in days): ");
```

```
    scanf("%d", &newMission.duration);
    getchar();

    printf("Enter Mission Type (Crew or Cargo): ");
    fgets(newMission.missionType, sizeof(newMission.missionType), stdin);
    newMission.missionType[strcspn(newMission.missionType, "\n")] = '\0';

    if (strcmp(newMission.missionType, "Crew") == 0) {
        manageCrewMission(&newMission);
    } else if (strcmp(newMission.missionType, "Cargo") == 0) {
        manageCargoMission(&newMission);
    } else {
        printf("Invalid mission type. It should be either 'Crew' or 'Cargo'.\n");
        return;
    }

    missions[*count] = newMission;
    (*count)++;
    totalMissions++;
}

void manageCrewMission(struct Mission *mission) {
    int crewCount;
    printf("Enter the number of crew members: ");
    scanf("%d", &crewCount);
    getchar();

    if (crewCount > MAX_CREW) {
```

```
        printf("Max crew members exceeded.\n");
        return;
    }

    for (int i = 0; i < crewCount; i++) {
        printf("Enter Crew Member %d Name: ", i + 1);
        fgets(mission->payload.crew[i].name, sizeof(mission->payload.crew[i].name),
stdin);
        mission->payload.crew[i].name[strcspn(mission->payload.crew[i].name, "\n")] =
'\0';

        printf("Enter Crew Member %d Age: ", i + 1);
        scanf("%d", &mission->payload.crew[i].age);
        getchar();

        printf("Enter Crew Member %d Role: ", i + 1);
        fgets(mission->payload.crew[i].role, sizeof(mission->payload.crew[i].role), stdin);
        mission->payload.crew[i].role[strcspn(mission->payload.crew[i].role, "\n")] = '\0';
    }
}

void manageCargoMission(struct Mission *mission) {
    int cargoCount;
    printf("Enter the number of cargo items: ");
    scanf("%d", &cargoCount);
    getchar();

    if (cargoCount > MAX_CARGO) {
        printf("Max cargo items exceeded.\n");
```

```
        return;
    }

    for (int i = 0; i < cargoCount; i++) {
        printf("Enter Cargo %d Name: ", i + 1);
        fgets(mission->payload.cargo[i].cargoName, sizeof(mission->payload.cargo[i].cargoName), stdin);
        mission->payload.cargo[i].cargoName[strcspn(mission->payload.cargo[i].cargoName, "\n")] = '\0';

        printf("Enter Cargo %d Weight: ", i + 1);
        scanf("%f", &mission->payload.cargo[i].weight);
        getchar();
    }
}

void updateMission(struct Mission *missions, int count) {
    int missionID;
    printf("Enter Mission ID to update: ");
    scanf("%d", &missionID);
    getchar();

    int found = 0;
    for (int i = 0; i < count; i++) {
        if (missions[i].missionID == missionID) {
            found = 1;
            printf("Enter new Mission Name: ");
            fgets(missions[i].name, sizeof(missions[i].name), stdin);
            missions[i].name[strcspn(missions[i].name, "\n")] = '\0';
```

```
            printf("Enter new Mission Duration (in days): ");
            scanf("%d", &missions[i].duration);
            getchar();

            printf("Enter new Mission Type (Crew or Cargo): ");
            fgets(missions[i].missionType, sizeof(missions[i].missionType), stdin);
            missions[i].missionType[strcspn(missions[i].missionType, "\n")] = '\0';

            if (strcmp(missions[i].missionType, "Crew") == 0) {
                manageCrewMission(&missions[i]);
            } else if (strcmp(missions[i].missionType, "Cargo") == 0) {
                manageCargoMission(&missions[i]);
            } else {
                printf("Invalid mission type. It should be either 'Crew' or 'Cargo'.\n");
            }
            break;
        }
    }

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

void displayMissionSummary(const struct Mission *missions, int count) {
    for (int i = 0; i < count; i++) {
        printf("\n----- Mission Summary -----\n");
```

```
        printf("Mission ID: %d\n", missions[i].missionID);
        printf("Mission Name: %s\n", missions[i].name);
        printf("Mission Duration: %d days\n", missions[i].duration);
        printf("Mission Type: %s\n", missions[i].missionType);

        if (strcmp(missions[i].missionType, "Crew") == 0) {
            printf("Crew Details:\n");
            for (int j = 0; j < MAX_CREW && strlen(missions[i].payload.crew[j].name) > 0; j++) {
                printf("Name: %s, Age: %d, Role: %s\n", missions[i].payload.crew[j].name,
missions[i].payload.crew[j].age, missions[i].payload.crew[j].role);
            }
        } else if (strcmp(missions[i].missionType, "Cargo") == 0) {
            printf("Cargo Details:\n");
            for (int j = 0; j < MAX_CARGO && strlen(missions[i].payload.cargo[j].cargoName) >
0; j++) {
                printf("Cargo: %s, Weight: %.2f kg\n", missions[i].payload.cargo[j].cargoName,
missions[i].payload.cargo[j].weight);
            }
        }
    }
}
```

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 MAX_LOGS 10
#define MAINTENANCE_FREQUENCY 6

struct RoutineMaintenance {
  char description[100];
  int hours;
};

struct EmergencyMaintenance {
  char description[100];
  int severity;
};

union MaintenanceType {
  struct RoutineMaintenance routine;
  struct EmergencyMaintenance emergency;
};
```

```
struct MaintenanceLog {
   int logID;
   char aircraftID[10];
   char date[20];
   union MaintenanceType maintenance;
   char maintenanceType[20];
};

static int totalLogs = 0;

void addLog(struct MaintenanceLog *logs, int *count);
void displayLogs(const struct MaintenanceLog *logs, int count);

int main() {
   struct MaintenanceLog logs[MAX_LOGS];
   int choice;
   int logCount = 0;

   while (1) {
      printf("\n----- Aircraft Maintenance Tracker -----\n");
      printf("1. Add Maintenance Log\n");
      printf("2. Display Maintenance Logs\n");
      printf("3. Exit\n");
      printf("Enter your choice: ");
      scanf("%d", &choice);
      getchar();

      switch (choice) {
```

```
        case 1:
            addLog(logs, &logCount);
            break;
        case 2:
            displayLogs(logs, logCount);
            break;
        case 3:
            printf("Exiting the system.\n");
            return 0;
        default:
            printf("Invalid choice. Please try again.\n");
      }
   }

   return 0;
}

void addLog(struct MaintenanceLog *logs, int *count) {
   if (*count >= MAX_LOGS) {
      printf("Max logs reached.\n");
      return;
   }

   struct MaintenanceLog newLog;
   printf("Enter Log ID: ");
   scanf("%d", &newLog.logID);
   getchar();
```

```
    printf("Enter Aircraft ID: ");
    fgets(newLog.aircraftID, sizeof(newLog.aircraftID), stdin);
    newLog.aircraftID[strcspn(newLog.aircraftID, "\n")] = '\0';

    printf("Enter Date (DD/MM/YYYY): ");
    fgets(newLog.date, sizeof(newLog.date), stdin);
    newLog.date[strcspn(newLog.date, "\n")] = '\0';

    printf("Enter Maintenance Type (Routine/Emergency): ");
    fgets(newLog.maintenanceType, sizeof(newLog.maintenanceType), stdin);
    newLog.maintenanceType[strcspn(newLog.maintenanceType, "\n")] = '\0';

    if (strcmp(newLog.maintenanceType, "Routine") == 0) {
        printf("Enter Routine Maintenance Description: ");
        fgets(newLog.maintenance.routine.description,
sizeof(newLog.maintenance.routine.description), stdin);
        newLog.maintenance.routine.description[strcspn(newLog.maintenance.routine.de
scription, "\n")] = '\0';
        printf("Enter Routine Maintenance Duration (hours): ");
        scanf("%d", &newLog.maintenance.routine.hours);
    } else if (strcmp(newLog.maintenanceType, "Emergency") == 0) {
        printf("Enter Emergency Maintenance Description: ");
        fgets(newLog.maintenance.emergency.description,
sizeof(newLog.maintenance.emergency.description), stdin);
        newLog.maintenance.emergency.description[strcspn(newLog.maintenance.emerg
ency.description, "\n")] = '\0';
        printf("Enter Emergency Severity (1-10): ");
        scanf("%d", &newLog.maintenance.emergency.severity);
    } else {
```

```
        printf("Invalid maintenance type.\n");
        return;
    }

    logs[*count] = newLog;
    (*count)++;
    totalLogs++;
}

void displayLogs(const struct MaintenanceLog *logs, int count) {
    for (int i = 0; i < count; i++) {
        printf("\n----- Maintenance Log -----\n");
        printf("Log ID: %d\n", logs[i].logID);
        printf("Aircraft ID: %s\n", logs[i].aircraftID);
        printf("Date: %s\n", logs[i].date);
        printf("Maintenance Type: %s\n", logs[i].maintenanceType);

        if (strcmp(logs[i].maintenanceType, "Routine") == 0) {
            printf("Routine Maintenance Description: %s\n",
logs[i].maintenance.routine.description);
            printf("Routine Maintenance Duration: %d hours\n",
logs[i].maintenance.routine.hours);
        } else if (strcmp(logs[i].maintenanceType, "Emergency") == 0) {
            printf("Emergency Maintenance Description: %s\n",
logs[i].maintenance.emergency.description);
            printf("Emergency Maintenance Severity: %d\n",
logs[i].maintenance.emergency.severity);
        }
    }
```

}

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>
#include <string.h>

#define MAX_UPDATES 10

struct ManualMode {
  float x, y, z;
};

struct AutomaticMode {
  float velocity, time;
};
```

```
union NavigationMode {
   struct ManualMode manual;
   struct AutomaticMode automatic;
};

struct NavigationData {
   float position[3];
   float velocity;
   union NavigationMode mode;
   char navigationMode[20];
};

static int totalUpdates = 0;

void updateNavigationData(struct NavigationData *data, int *count);
void displayNavigationStatus(const struct NavigationData *data, int count);

int main() {
   struct NavigationData data[MAX_UPDATES];
   int choice;
   int updateCount = 0;

   while (1) {
      printf("\n----- Spacecraft Navigation System -----\n");
      printf("1. Update Navigation Data\n");
      printf("2. Display Navigation Status\n");
      printf("3. Exit\n");
      printf("Enter your choice: ");
```

```
        scanf("%d", &choice);
        getchar();

        switch (choice) {
            case 1:
                updateNavigationData(data, &updateCount);
                break;
            case 2:
                displayNavigationStatus(data, updateCount);
                break;
            case 3:
                printf("Exiting the system.\n");
                return 0;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}

void updateNavigationData(struct NavigationData *data, int *count) {
    if (*count >= MAX_UPDATES) {
        printf("Max updates reached.\n");
        return;
    }

    struct NavigationData newData;
```

```
    printf("Enter Navigation Mode (Manual/Automatic): ");
    fgets(newData.navigationMode, sizeof(newData.navigationMode), stdin);
    newData.navigationMode[strcspn(newData.navigationMode, "\n")] = '\0';

    printf("Enter Position (x y z): ");
    scanf("%f %f %f", &newData.position[0], &newData.position[1],
&newData.position[2]);

    if (strcmp(newData.navigationMode, "Manual") == 0) {
        printf("Enter Manual Mode Coordinates (x, y, z): ");
        scanf("%f %f %f", &newData.mode.manual.x, &newData.mode.manual.y,
&newData.mode.manual.z);
    } else if (strcmp(newData.navigationMode, "Automatic") == 0) {
        printf("Enter Automatic Mode Velocity and Time: ");
        scanf("%f %f", &newData.mode.automatic.velocity,
&newData.mode.automatic.time);
    } else {
        printf("Invalid navigation mode.\n");
        return;
    }

    data[*count] = newData;
    (*count)++;
    totalUpdates++;
}

void displayNavigationStatus(const struct NavigationData *data, int count) {
    for (int i = 0; i < count; i++) {
        printf("\n----- Navigation Status -----\n");
```

```
    printf("Position: (%.2f, %.2f, %.2f)\n", data[i].position[0], data[i].position[1],
data[i].position[2]);

    printf("Navigation Mode: %s\n", data[i].navigationMode);


    if (strcmp(data[i].navigationMode, "Manual") == 0) {

      printf("Manual Coordinates: (%.2f, %.2f, %.2f)\n", data[i].mode.manual.x,
data[i].mode.manual.y, data[i].mode.manual.z);

    } else if (strcmp(data[i].navigationMode, "Automatic") == 0) {

      printf("Automatic Mode Velocity: %.2f, Time: %.2f\n",
data[i].mode.automatic.velocity, data[i].mode.automatic.time);

    }

  }

}
```

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>
```

```
#define MAX_SIMULATIONS 5

struct ManualControl {
  float speed;
  float altitude;
};

struct AutomatedControl {
  float autopilotSpeed;
  float autopilotAltitude;
};

union ControlSettings {
  struct ManualControl manual;
  struct AutomatedControl automated;
};

struct Simulation {
  int simulationID;
  char aircraftModel[50];
  int duration;
  union ControlSettings control;
  char controlMode[20];
};

const int fixedDuration = 3;
```

```
void startSimulation(struct Simulation *simulations, int *count);
void displaySimulationResults(const struct Simulation *simulations, int count);

int main() {
    struct Simulation simulations[MAX_SIMULATIONS];
    int choice;
    int simulationCount = 0;

    while (1) {
        printf("\n----- Flight Simulation Control -----\n");
        printf("1. Start Simulation\n");
        printf("2. Display Simulation Results\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        getchar();

        switch (choice) {
            case 1:
                startSimulation(simulations, &simulationCount);
                break;
            case 2:
                displaySimulationResults(simulations, simulationCount);
                break;
            case 3:
                printf("Exiting the system.\n");
                return 0;
            default:
```

```
            printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}

void startSimulation(struct Simulation *simulations, int *count) {
    if (*count >= MAX_SIMULATIONS) {
        printf("Max simulations reached.\n");
        return;
    }

    struct Simulation newSim;
    printf("Enter Simulation ID: ");
    scanf("%d", &newSim.simulationID);
    getchar();

    printf("Enter Aircraft Model: ");
    fgets(newSim.aircraftModel, sizeof(newSim.aircraftModel), stdin);
    newSim.aircraftModel[strcspn(newSim.aircraftModel, "\n")] = '\0';

    printf("Enter Duration (hours): ");
    scanf("%d", &newSim.duration);

    printf("Enter Control Mode (Manual/Automated): ");
    fgets(newSim.controlMode, sizeof(newSim.controlMode), stdin);
    newSim.controlMode[strcspn(newSim.controlMode, "\n")] = '\0';
```

```
    if (strcmp(newSim.controlMode, "Manual") == 0) {
        printf("Enter Speed and Altitude: ");
        scanf("%f %f", &newSim.control.manual.speed, &newSim.control.manual.altitude);
    } else if (strcmp(newSim.controlMode, "Automated") == 0) {
        printf("Enter Autopilot Speed and Altitude: ");
        scanf("%f %f", &newSim.control.automated.autopilotSpeed,
&newSim.control.automated.autopilotAltitude);
    } else {
        printf("Invalid control mode.\n");
        return;
    }

    simulations[*count] = newSim;
    (*count)++;
}

void displaySimulationResults(const struct Simulation *simulations, int count) {
    for (int i = 0; i < count; i++) {
        printf("\n----- Simulation Result -----\n");
        printf("Simulation ID: %d\n", simulations[i].simulationID);
        printf("Aircraft Model: %s\n", simulations[i].aircraftModel);
        printf("Duration: %d hours\n", simulations[i].duration);
        printf("Control Mode: %s\n", simulations[i].controlMode);

        if (strcmp(simulations[i].controlMode, "Manual") == 0) {
            printf("Manual Control Speed: %.2f, Altitude: %.2f\n",
simulations[i].control.manual.speed, simulations[i].control.manual.altitude);
        } else if (strcmp(simulations[i].controlMode, "Automated") == 0) {
```

```
        printf("Automated Control Autopilot Speed: %.2f, Altitude: %.2f\n",
simulations[i].control.automated.autopilotSpeed,
simulations[i].control.automated.autopilotAltitude);
    }
  }
}
```

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>
#include <string.h>

#define MAX_TESTS 100

struct ComponentTest {
```

```
    int testID;
    char componentName[50];
    union TestData {
        float physicalParameter; // e.g., weight, pressure
        char softwareVersion[50];
    } testData;
    int testType; // 1 for physical, 2 for software
};

static int totalTests = 0;

void recordTest(struct ComponentTest *tests, int *testCount) {
    if (*testCount >= MAX_TESTS) {
        printf("Max test records reached.\n");
        return;
    }
    struct ComponentTest newTest;
    printf("Enter Test ID: ");
    scanf("%d", &newTest.testID);
    getchar();
    printf("Enter Component Name: ");
    fgets(newTest.componentName, sizeof(newTest.componentName), stdin);
    newTest.componentName[strcspn(newTest.componentName, "\n")] = '\0';
    printf("Enter Test Type (1-Physical, 2-Software): ");
    scanf("%d", &newTest.testType);

    if (newTest.testType == 1) {
        printf("Enter Physical Parameter (e.g., weight): ");
```

```
        scanf("%f", &newTest.testData.physicalParameter);
    } else if (newTest.testType == 2) {
        getchar();
        printf("Enter Software Version: ");
        fgets(newTest.testData.softwareVersion, sizeof(newTest.testData.softwareVersion),
stdin);
        newTest.testData.softwareVersion[strcspn(newTest.testData.softwareVersion, "\n")]
= '\0';
    } else {
        printf("Invalid test type.\n");
        return;
    }

    tests[*testCount] = newTest;
    (*testCount)++;
    totalTests++;
    printf("Test recorded successfully.\n");
}

void displayTests(const struct ComponentTest *tests, int testCount) {
    for (int i = 0; i < testCount; i++) {
        printf("\nTest ID: %d\n", tests[i].testID);
        printf("Component Name: %s\n", tests[i].componentName);
        if (tests[i].testType == 1) {
            printf("Test Type: Physical\n");
            printf("Physical Parameter: %.2f\n", tests[i].testData.physicalParameter);
        } else {
            printf("Test Type: Software\n");
            printf("Software Version: %s\n", tests[i].testData.softwareVersion);
```

```
    }
  }
  printf("Total Tests Conducted: %d\n", totalTests);
}
```

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>

#define MAX_CREW 10

struct CrewMember {
  int crewID;
  char name[50];
  char role[20];
  union RoleDetails {
```

```
        char engineeringSpecialization[50];
        char scientificField[50];
    } details;
};

void addCrewMember(struct CrewMember *crew, int *crewCount) {
    if (*crewCount >= MAX_CREW) {
        printf("Max crew limit reached.\n");
        return;
    }
    struct CrewMember newMember;
    printf("Enter Crew ID: ");
    scanf("%d", &newMember.crewID);
    getchar();
    printf("Enter Name: ");
    fgets(newMember.name, sizeof(newMember.name), stdin);
    newMember.name[strcspn(newMember.name, "\n")] = '\0';
    printf("Enter Role (Engineer/Scientist): ");
    fgets(newMember.role, sizeof(newMember.role), stdin);
    newMember.role[strcspn(newMember.role, "\n")] = '\0';

    if (strcmp(newMember.role, "Engineer") == 0) {
        printf("Enter Engineering Specialization: ");
        fgets(newMember.details.engineeringSpecialization,
sizeof(newMember.details.engineeringSpecialization), stdin);
        newMember.details.engineeringSpecialization[strcspn(newMember.details.enginee
ringSpecialization, "\n")] = '\0';
    } else if (strcmp(newMember.role, "Scientist") == 0) {
        printf("Enter Scientific Field: ");
```

```
        fgets(newMember.details.scientificField, sizeof(newMember.details.scientificField),
stdin);
        newMember.details.scientificField[strcspn(newMember.details.scientificField,
"\n")] = '\0';
    } else {
        printf("Invalid role.\n");
        return;
    }

    crew[*crewCount] = newMember;
    (*crewCount)++;
    printf("Crew member added successfully.\n");
}

void displayCrew(const struct CrewMember *crew, int crewCount) {
    for (int i = 0; i < crewCount; i++) {
        printf("\nCrew ID: %d\n", crew[i].crewID);
        printf("Name: %s\n", crew[i].name);
        printf("Role: %s\n", crew[i].role);
        if (strcmp(crew[i].role, "Engineer") == 0) {
            printf("Engineering Specialization: %s\n",
crew[i].details.engineeringSpecialization);
        } else {
            printf("Scientific Field: %s\n", crew[i].details.scientificField);
        }
    }
}
```

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>

#define MAX_DATA 100

struct ResearchData {
  int experimentID;
  char description[100];
  union DataType {
    float numericalData;
    char qualitativeData[50];
  } data;
  int dataType; // 1 for numerical, 2 for qualitative
};

static int totalAnalyses = 0;
```

```
void analyzeData(struct ResearchData *dataRecords, int *dataCount) {
    struct ResearchData newData;
    printf("Enter Experiment ID: ");
    scanf("%d", &newData.experimentID);
    getchar();
    printf("Enter Description: ");
    fgets(newData.description, sizeof(newData.description), stdin);
    newData.description[strcspn(newData.description, "\n")] = '\0';
    printf("Enter Data Type (1-Numerical, 2-Qualitative): ");
    scanf("%d", &newData.dataType);

    if (newData.dataType == 1) {
        printf("Enter Numerical Data: ");
        scanf("%f", &newData.data.numericalData);
    } else if (newData.dataType == 2) {
        getchar();
        printf("Enter Qualitative Data: ");
        fgets(newData.data.qualitativeData, sizeof(newData.data.qualitativeData), stdin);
        newData.data.qualitativeData[strcspn(newData.data.qualitativeData, "\n")] = '\0';
    } else {
        printf("Invalid data type.\n");
        return;
    }

    dataRecords[*dataCount] = newData;
    (*dataCount)++;
    totalAnalyses++;
```

```
    printf("Data analyzed successfully.\n");
}

void displayReports(const struct ResearchData *dataRecords, int dataCount) {
    for (int i = 0; i < dataCount; i++) {
        printf("\nExperiment ID: %d\n", dataRecords[i].experimentID);
        printf("Description: %s\n", dataRecords[i].description);
        if (dataRecords[i].dataType == 1) {
            printf("Numerical Data: %.2f\n", dataRecords[i].data.numericalData);
        } else {
            printf("Qualitative Data: %s\n", dataRecords[i].data.qualitativeData);
        }
    }
    printf("Total Analyses Conducted: %d\n", totalAnalyses);
}
```

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>

#define MAX_LAUNCHES 50

struct Launch {
   int launchID;
   char rocketName[50];
   char date[20];
   union Status {
      char scheduled[20];
      char completed[20];
   } status;
   int statusType; // 1 for scheduled, 2 for completed
};

// Function to schedule a launch (call by reference)
void scheduleLaunch(struct Launch *launches, int *launchCount) {
   if (*launchCount >= MAX_LAUNCHES) {
      printf("Maximum launch limit reached.\n");
      return;
   }

   struct Launch newLaunch;
   printf("Enter Launch ID: ");
   scanf("%d", &newLaunch.launchID);
   getchar(); // Clear newline character from buffer
```

```
    printf("Enter Rocket Name: ");
    fgets(newLaunch.rocketName, sizeof(newLaunch.rocketName), stdin);
    newLaunch.rocketName[strcspn(newLaunch.rocketName, "\n")] = '\0';

    printf("Enter Launch Date (DD/MM/YYYY): ");
    fgets(newLaunch.date, sizeof(newLaunch.date), stdin);
    newLaunch.date[strcspn(newLaunch.date, "\n")] = '\0';

    printf("Enter Status Type (1 - Scheduled, 2 - Completed): ");
    scanf("%d", &newLaunch.statusType);

    if (newLaunch.statusType == 1) {
        strcpy(newLaunch.status.scheduled, "Scheduled");
    } else if (newLaunch.statusType == 2) {
        strcpy(newLaunch.status.completed, "Completed");
    } else {
        printf("Invalid status type. Launch not added.\n");
        return;
    }

    launches[*launchCount] = newLaunch;
    (*launchCount)++;
    printf("Launch scheduled successfully.\n");
}

// Function to update launch status (call by reference)
void updateLaunchStatus(struct Launch *launches, int launchCount) {
```

```
int launchID;
printf("Enter Launch ID to update status: ");
scanf("%d", &launchID);

for (int i = 0; i < launchCount; i++) {
   if (launches[i].launchID == launchID) {
      printf("Current Status: %s\n",
           launches[i].statusType == 1 ? "Scheduled" : "Completed");

      printf("Enter New Status (1 - Scheduled, 2 - Completed): ");
      int newStatus;
      scanf("%d", &newStatus);

      if (newStatus == 1) {
         strcpy(launches[i].status.scheduled, "Scheduled");
         launches[i].statusType = 1;
      } else if (newStatus == 2) {
         strcpy(launches[i].status.completed, "Completed");
         launches[i].statusType = 2;
      } else {
         printf("Invalid status type. No changes made.\n");
         return;
      }

      printf("Launch status updated successfully.\n");
      return;
   }
}
```

```
    printf("Launch ID not found.\n");
}

// Function to display launch schedules (call by value)
void displayLaunches(const struct Launch *launches, int launchCount) {
    if (launchCount == 0) {
        printf("No launches scheduled.\n");
        return;
    }

    printf("\nLaunch Schedule:\n");
    for (int i = 0; i < launchCount; i++) {
        printf("\nLaunch ID: %d\n", launches[i].launchID);
        printf("Rocket Name: %s\n", launches[i].rocketName);
        printf("Launch Date: %s\n", launches[i].date);
        printf("Status: %s\n", launches[i].statusType == 1 ? "Scheduled" : "Completed");
    }
}

int main() {
    struct Launch launches[MAX_LAUNCHES];
    int launchCount = 0;
    int choice;

    do {
        printf("\nRocket Launch Scheduler\n");
        printf("1. Schedule a Launch\n");
        printf("2. Update Launch Status\n");
```

```
        printf("3. Display Launch Schedules\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
        case 1:
            scheduleLaunch(launches, &launchCount);
            break;
        case 2:
            updateLaunchStatus(launches, launchCount);
            break;
        case 3:
            displayLaunches(launches, launchCount);
            break;
        case 4:
            printf("Exiting...\n");
            break;
        default:
            printf("Invalid choice. Please try again.\n");
        }
    } while (choice != 4);

    return 0;
}
```