```c
/*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>
struct item
{
    char itemid[10];
    char itemname[20];
    float price;
    int quantity;
};
void Add(struct item *i,int count)
{
        printf("Enter itemid : ");
        scanf("%s",i[count].itemid);
        printf("Enter item name : ");
        scanf("%s",i[count].itemname);
        printf("Enter price : ");
        scanf("%f",&i[count].price);
        printf("Enter quantity : ");
        scanf("%d",&i[count].quantity);
}
void Update(struct item *i,char id[],int count)
{
    for(int j=0;j<count+1;j++)
    {
        if(strcmp(i[j].itemid,id)==0)
```

```
        {
            printf("Enter price : ");
        scanf("%f",&i[j].price);
        printf("Enter quantity : ");
        scanf("%d",&i[j].quantity);
        }
    }
}
void Display(struct item *i,int count)
{
    printf("\nStock Details\n");
    for(int j=0;j<count;j++)
    {
        printf("Itemid : %s\n",i[j].itemid);
        printf("Item name : %s\n",i[j].itemname);
        printf("Price : %.2f\n",i[j].price);
        printf("Quantity : %d\n",i[j].quantity);
    }
}
void main()
{
struct item i[5];
int choice,count=0;
char id[20];
do
{
    printf("\n1.Add item\n");
    printf("2.Update details\n");
    printf("3.Exit\n");
    printf("Enter choice : ");
    scanf("%d",&choice);
    switch (choice)
    {
    case 1:
        Add(i,count);
        count++;
        Display(i,count);
        break;
    case 2:
    printf("Enter item id to update : ");
```

```
    scanf("%s",id);
        Update(i,id,count);
        Display(i,count);
        break;
    case 3:printf("Exiting....\n");
    break;

    default:printf("Enter valid input\n");
        break;
    }

} while (choice!=3);

}
```

```
1.Add item
2.Update details
3.Exit
Enter choice : 1
Enter itemid : i1
Enter item name : n1
Enter price : 20
Enter quantity : 20

Stock Details
Itemid : i1
Item name : n1
Price : 20.00
Quantity : 20

1.Add item
2.Update details
3.Exit
Enter choice : 2
Enter item id to update : i1
Enter price : 400
Enter quantity : 30

Stock Details
Itemid : i1
Item name : n1
Price : 400.00
Quantity : 30

1.Add item
2.Update details
3.Exit
Enter choice :
```

```c
/*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 DISCOUNT_RATE 0.1
struct Item {
    char itemName[30];
    float price;
    int quantity;
};
struct Order {
    char orderID[10];
    char customerName[50];
    struct Item items[5];
    int itemCount;
    float totalCost;
};

float calculateTotalCost(struct Order order) {
    float total = 0;
    for (int i = 0; i < order.itemCount; i++) {
        total += order.items[i].price * order.items[i].quantity;
    }
    return total;
}
void processOrders(struct Order orders[], int orderCount) {
    for (int i = 0; i < orderCount; i++) {
        orders[i].totalCost = calculateTotalCost(orders[i]);
        printf("\nOrder ID: %s\n", orders[i].orderID);
        printf("Customer Name: %s\n", orders[i].customerName);
```

```
        printf("Total Cost (Before Discount): %.2f\n",
orders[i].totalCost);
        float discountedCost = orders[i].totalCost - (orders[i].totalCost
* DISCOUNT_RATE);
        printf("Total Cost (After Discount): %.2f\n", discountedCost);
    }
}

int main() {
    int orderCount;
    printf("Enter number of orders: ");
    scanf("%d", &orderCount);
    struct Order orders[orderCount];
    for (int i = 0; i < orderCount; i++) {
        printf("\nOrder %d:\n", i + 1);
        printf("Enter Order ID: ");
        scanf("%s", orders[i].orderID);
        printf("Enter Customer Name: ");
        scanf(" %[^\n]s", orders[i].customerName);
        printf("Enter the number of items: ");
        scanf("%d", &orders[i].itemCount);
        for (int j = 0; j < orders[i].itemCount; j++) {
            printf("Item %d:\n", j + 1);
            printf("Enter Item Name: ");
            scanf("%s", orders[i].items[j].itemName);
            printf("Enter Item Price: ");
            scanf("%f", &orders[i].items[j].price);
            printf("Enter Item Quantity: ");
            scanf("%d", &orders[i].items[j].quantity);
        }
    }
    processOrders(orders, orderCount);
}
```

```
Order ID: i3
Customer Name: c3
Total Cost (Before Discount): 200.00
Total Cost (After Discount): 180.00
PS D:\projects\quest\C>
```

```c
/*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_FEEDBACK 100
struct Feedback
{
    int customerID;
    char feedbackText[200];
    int rating;
};
void addFeedback(struct Feedback feedbacks[], int *count)
{
    if (*count >= MAX_FEEDBACK) {
        printf("Maximum feedback limit reached.\n");
        return;
    }
    printf("Enter Customer ID: ");
    scanf("%d", &feedbacks[*count].customerID);
    printf("Enter Feedback Text: ");
    getchar();
    fgets(feedbacks[*count].feedbackText,
sizeof(feedbacks[*count].feedbackText), stdin);
```

```
    feedbacks[*count].feedbackText[strcspn(feedbacks[*count].feedbackText,
"\n")] = '\0';

    do {
        printf("Enter Rating (1-5): ");
        scanf("%d", &feedbacks[*count].rating);
        if (feedbacks[*count].rating < 1 || feedbacks[*count].rating > 5)
        {
            printf("Invalid rating. Please enter a rating between 1 and
5.\n");
        }
    } while (feedbacks[*count].rating < 1 || feedbacks[*count].rating >
5);

    (*count)++;
    printf("Feedback added successfully!\n");
}
void displayFeedbackSummary(struct Feedback feedbacks[], int count)
{

    if (count == 0) {
        printf("No feedback available.\n");
        return;
    }

    printf("\nFeedback Summary:\n");
    for (int i = 0; i < count; i++)
    {
        printf("\nCustomer ID: %d\n", feedbacks[i].customerID);
        printf("Feedback: %s\n", feedbacks[i].feedbackText);
        printf("Rating: %d - ", feedbacks[i].rating);
        categorizeFeedback(feedbacks[i].rating);
    }
}
void categorizeFeedback(int rating) {
    switch (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() {
    struct Feedback feedbacks[MAX_FEEDBACK];
    int count = 0;
    int choice;

    do {
        printf("\nFeedback Management System:\n");
        printf("1. Add Feedback\n");
        printf("2. Display Feedback Summary\n");
        printf("3. 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. Please try again.\n");
        }
    } while (choice != 3);

    return 0;
}
```

```
Feedback Management System:
1. Add Feedback
2. Display Feedback Summary
3. Exit
Enter your choice: 1
Enter Customer ID: i2
Enter Feedback Text: Enter Rating (1-5): 2
Feedback added successfully!

Feedback Management System:
1. Add Feedback
2. Display Feedback Summary
3. Exit
Enter your choice: 2

Feedback Summary:

Customer ID: 0
Feedback: 1
Rating: 3 - Average

Customer ID: 0
Feedback: 2
Rating: 2 - Poor
```

```
/*: 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.
```

```c
/*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>
#define card .13
#define upi .12
#define cash 0
struct payment
{
    float amount;
    char type[20];
    float charge;
};
void add(struct payment *p)
{
    const float cd=card,u=upi,c=cash;
    printf("Enter amount : ");
    scanf("%f",&p->amount);
    printf("Enter method : ");
    scanf("%s",p->type);
    if(strcmp(p->type,"card")==0)
    p->charge=p->amount*cd;
    else if(strcmp(p->type,"upi")==0)
    p->charge=p->amount*u;
     else if(strcmp(p->type,"cash")==0)
     p->charge=p->amount*c;
     else
    { printf("Invalid transaction type\n");
     p->charge=0;
    }
}
void display(struct payment p)
{
    printf("Payment amount : %f\n",p.amount);
    printf("Payment type : %s\n",p.type);
```

```
    printf("Payment charge : %f\n",p.charge);
}
void main()
{
struct payment payments[5];
for(int i=0;i<5;i++)
{
    add(&payments[i]);
}
printf("Payment details\n");
for (int i = 0; i < 5; i++)
{
  display(payments[i]);
}

}
```

```
PS D:\projects\quest\C> cd "d:\project
Enter amount : 200
Enter method : upi
Enter amount : 500
Enter method : cash
Enter amount : 2000
Enter method : card
Enter amount : 10000
Enter method : upi
Enter amount : 2000
Enter method : cash
Payment details
Payment amount : 200.000000
Payment type : upi
Payment charge : 24.000000
Payment amount : 500.000000
Payment type : cash
Payment charge : 0.000000
Payment amount : 2000.000000
Payment type : card
Payment charge : 260.000000
Payment amount : 10000.000000
Payment type : upi
Payment charge : 1200.000000
Payment amount : 2000.000000
Payment type : cash
Payment charge : 0.000000
```

```
/*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:
```

```c
Display the updated cart after each operation.
 */
#include<stdio.h>
#include<string.h>

struct cart {
    char itemid[10];
    char itemname[20];
    float price;
    int quantity;
};
void add(struct cart *c)
{
    printf("Enter item id: ");
    scanf("%s", c->itemid);
    printf("Enter item name: ");
    scanf("%s", c->itemname);
    printf("Enter price: ");
    scanf("%f", &c->price);
    printf("Enter quantity: ");
    scanf("%d", &c->quantity);
}

void delete(struct cart *c, char id[])
{
    int q;
    for (int i = 0; i < 5; i++) {
        if (strcmp(c[i].itemid, id) == 0)
        {
            printf("Enter quantity to remove: ");
            scanf("%d", &q);
            if (q <= c[i].quantity)
            {
                c[i].quantity -= q;
                printf("Removed %d items. New quantity: %d\n", q,
c[i].quantity);
            }
            else
            printf("Not enough quantity to remove. Available quantity:
%d\n", c[i].quantity);
```

```
            return;
        }
    }
    printf("Item with id %s not found.\n", id);
}

void display(struct cart *c, char id[])
{
    for (int i = 0; i < 5; i++)
    {
        if (strcmp(c[i].itemid, id) == 0)
        {
            printf("Item name: %s\n", c[i].itemname);
            printf("Item price: %.2f\n", c[i].price);
            printf("Item quantity: %d\n", c[i].quantity);
            return;
        }
    }
    printf("Item with id %s not found.\n", id);
}

void main()
{
    struct cart c[5];
    int choice;
    char id[10];
    for (int i = 0; i < 5; i++)
    c[i].quantity = 0;

    do {
        printf("\n1. Add\n");
        printf("2. Remove\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
```

```
        case 1:
            for (int i = 0; i < 5; i++)
                add(&c[i]);

            break;
        case 2:
            printf("Enter item id to remove: ");
            scanf("%s", id);
            delete(c, id);
            break;
        case 3:
            printf("Enter item id to display: ");
            scanf("%s", id);
            display(c, id);
            break;
        case 4:
            printf("Exiting program.\n");
            break;
        default:
            printf("Invalid choice! Please try again.\n");
            break;
        }
    } while (choice != 4);
}
```

```
Enter quantity: 20

1. Add
2. Remove
3. Display
4. Exit
Enter choice: 3
Enter item id to display: i1
Item name: n1
Item price: 20.00
Item quantity: 20

1. Add
2. Remove
3. Display
4. Exit
Enter choice: 2
Enter item id to remove: i1
Enter quantity to remove: 10
Removed 10 items. New quantity: 10

1. Add
2. Remove
3. Display
4. Exit
Enter choice: 3
Enter item id to display: i1
Item name: n1
Item price: 20.00
Item quantity: 10
```

```
/*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:
```

```c
Display product details if found or a message indicating the product is
not found.*/
#include <stdio.h>
#include <string.h>
struct product {
    char productid[10];
    char prductname[10];
    char category[10];
    float price;
};
void add(struct product *p) {
    printf("Enter product ID: ");
    scanf("%s", p->productid);
    printf("Enter product name: ");
    scanf("%s", p->prductname);
    printf("Enter category: ");
    scanf("%s", p->category);
    printf("Enter price: ");
    scanf("%f", &p->price);
}
int search(struct product *p, int size, char name[]) {
    for (int i = 0; i < size; i++) {
        if (strcmp(p[i].prductname, name) == 0) {
            return i;
        }
    }
    return -1;
}
void update(struct product *p) {
    printf("Enter updated price: ");
    scanf("%f", &p->price);
    printf("Product price updated successfully.\n");
}
void display(struct product p) {
    printf("Product ID: %s\n", p.productid);
    printf("Product Name: %s\n", p.prductname);
    printf("Category: %s\n", p.category);
    printf("Price: %.2f\n", p.price);
}
int main() {
```

```
    struct product products[5];
    char name[10];
    int index;
    for (int i = 0; i < 5; i++) {
        printf("\nAdding product %d:\n", i + 1);
        add(&products[i]);
    }
    printf("\nEnter product name to search: ");
    scanf("%s", name);
    index = search(products, 5, name);

    if (index != -1) {
        printf("\nProduct found:\n");
        display(products[index]);
        update(&products[index]);
        printf("\nUpdated product details:\n");
        display(products[index]);
    } else {
        printf("\nProduct with name '%s' not found.\n", name);
    }

}
```

```
Adding product 2:
Enter product ID: i2
Enter product name: n2
Enter category: bat
Enter price: 30

Adding product 3:
Enter product ID: i3
Enter product name: n3
Enter category: ball
Enter price: 25

Adding product 4:
Enter product ID: i4
Enter product name: n4
Enter category: bat
Enter price: 50

Adding product 5:
Enter product ID: i5
Enter product name: n5
Enter category: bat
Enter price: 100

Enter product name to search: n2

Product found:
Product ID: i2
Product Name: n2
Category: bat
Price: 30.00
Enter updated price: 25
Product price updated successfully.

Updated product details:
Product ID: i2
Product Name: n2
Category: bat
Price: 25.00
PS D:\projects\quest\C>
```

```c
/*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.*/
#include <stdio.h>
#include <string.h>
struct Sale {
    int saleID;
    char productCategory[20];
    float amount;
    char date[11];
};
void addSale(struct Sale *sale)
{
    printf("Enter Sale ID: ");
    scanf("%d", &sale->saleID);
    printf("Enter Product Category: ");
    scanf("%s", sale->productCategory);
    printf("Enter Amount: ");
    scanf("%f", &sale->amount);
    printf("Enter Date (YYYY-MM-DD): ");
    scanf("%s", sale->date);
}
void generateReport(struct Sale *sales, int size)
{
    float electronicsTotal = 0, groceriesTotal = 0, clothingTotal = 0,
othersTotal = 0;
    for (int i = 0; i < size; i++)
    {
        if (strcmp(sales[i].productCategory, "Electronics") == 0)
        electronicsTotal += sales[i].amount;
        else if (strcmp(sales[i].productCategory, "Groceries") == 0)
        groceriesTotal += sales[i].amount;
         else if (strcmp(sales[i].productCategory, "Clothing") == 0)
        clothingTotal += sales[i].amount;
        else
```

```
        othersTotal += sales[i].amount;

    }
    printf("\nSales Report:\n");
    printf("Electronics : %.2f\n", electronicsTotal);
    printf("Groceries : %.2f\n", groceriesTotal);
    printf("Clothing : %.2f\n", clothingTotal);
    printf("Others : %.2f\n", othersTotal);
}
void main()
{
    struct Sale sales[100];
    int choice, count = 0;
    do {
        printf("\nSales Management System:\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:
                if (count < 100) {
                    printf("\nAdding Sale %d:\n", count + 1);
                    addSale(&sales[count]);
                    count++;
                }
                else
                printf("Sales array is full. Cannot add more sales.\n");
                break;
            case 2:
                if (count > 0)
                generateReport(sales, count);
                else
                printf("No sales data available to generate a report.\n");
                break;
            case 3:
                printf("Exiting...\n");
                break;
```

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

```
Sales Management System:
1. Add Sale
2. Generate Sales Report
3. Exit
Enter your choice: 1

Adding Sale 1:
Enter Sale ID: 1
Enter Product Category: bat
Enter Amount: 200
Enter Date (YYYY-MM-DD): 2005-01-10

Sales Management System:
1. Add Sale
2. Generate Sales Report
3. Exit
Enter your choice: 1

Adding Sale 2:
Enter Sale ID: 2
Enter Product Category: ball
Enter Amount: 500
Enter Date (YYYY-MM-DD): 2005-01-11

Sales Management System:
1. Add Sale
2. Generate Sales Report
3. Exit
Enter your choice: 2

Sales Report:
Electronics : 0.00
Groceries : 0.00
Clothing : 0.00
Others : 700.00

Sales Management System:
1. Add Sale
2. Generate Sales Report
3. Exit
```

```c
/*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 REWARD_RATE 0.1
struct Customer {
    int customerID;
    char name[50];
    float totalPurchases;
    int rewardPoints;
};
void addCustomer(struct Customer *c) {
    printf("Enter Customer ID: ");
    scanf("%d", &c->customerID);
    printf("Enter Customer Name: ");
    scanf(" %[^\n]s", c->name);
    printf("Enter Total Purchases: ");
    scanf("%f", &c->totalPurchases);
    c->rewardPoints = 0;
}
void updateRewardPoints(struct Customer *c)
{
    c->rewardPoints = (int)(c->totalPurchases * REWARD_RATE);
}
void displayCustomer(const struct Customer c)
{
    printf("\nCustomer Details:\n");
    printf("ID: %d\n", c.customerID);
    printf("Name: %s\n", c.name);
    printf("Total Purchases: $%.2f\n", c.totalPurchases);
    printf("Reward Points: %d\n", c.rewardPoints);
```

```
}
void main()
{

    struct Customer customers[5];
    for (int i = 0; i < 5; i++)
    {
        printf("\nEnter details for Customer %d:\n", i + 1);
        addCustomer(&customers[i]);
        updateRewardPoints(&customers[i]);
    }
    printf("\nCustomer Rewards Summary:\n");
    for (int i = 0; i < 5; i++)
    displayCustomer(customers[i]);

}
```

```
Customer Rewards Summary:

Customer Details:
ID: 1
Name: c1
Total Purchases: $2.00
Reward Points: 0

Customer Details:
ID: 2
Name: c2
Total Purchases: $10.00
Reward Points: 1

Customer Details:
ID: 3
Name: c3
Total Purchases: $40.00
Reward Points: 4

Customer Details:
ID: 4
Name: c4
Total Purchases: $500.00
Reward Points: 50

Customer Details:
ID: 5
Name: c5
Total Purchases: $600.00
Reward Points: 60
PS D:\projects\quest\C>
```

```
/*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.
```

```c
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>
struct WarehouseItem {
    int itemID;
    char itemName[50];
    int currentStock;
    int reorderLevel;
};
void addItem(struct WarehouseItem *item) {
    printf("Enter Item ID: ");
    scanf("%d", &item->itemID);
    printf("Enter Item Name: ");
    scanf(" %s", item->itemName);
    printf("Enter Current Stock: ");
    scanf("%d", &item->currentStock);
    printf("Enter Reorder Level: ");
    scanf("%d", &item->reorderLevel);
}
void updateStock(struct WarehouseItem *item)
{
    int additionalStock;
    printf("\nUpdate Stock for Item ID %d (%s):\n", item->itemID,
item->itemName);
    printf("Enter additional stock to add: ");
    scanf("%d", &additionalStock);
    item->currentStock += additionalStock;
}
void checkReorderStatus(const struct WarehouseItem item)
{
    printf("\nItem ID: %d\n", item.itemID);
    printf("Item Name: %s\n", item.itemName);
    printf("Current Stock: %d\n", item.currentStock);
    printf("Reorder Level: %d\n", item.reorderLevel);

    if (item.currentStock <= item.reorderLevel)
```

```
        printf("Status: Reorder Needed\n");
    else
        printf("Status: Stock Sufficient\n");

}
int main()
{
    int n;
    printf("Enter the number of items in the warehouse: ");
    scanf("%d", &n);
    struct WarehouseItem items[n];
    for (int i = 0; i < n; i++)
    {
        printf("\nEnter details for Item %d:\n", i + 1);
        addItem(&items[i]);
    }
    for (int i = 0; i < n; i++)
        updateStock(&items[i]);
    printf("\nWarehouse Stock Report:\n");
    for (int i = 0; i < n; i++)
        checkReorderStatus(items[i]);
}
```

```
Enter Reorder Level: 10

Update Stock for Item ID 1 (ball):
Enter additional stock to add: 10

Update Stock for Item ID 2 (bat):
Enter additional stock to add: 20

Update Stock for Item ID 3 (car):
Enter additional stock to add: 60

Update Stock for Item ID 4 (racket):
Enter additional stock to add: 50

Warehouse Stock Report:

Item ID: 1
Item Name: ball
Current Stock: 210
Reorder Level: 20
Status: Stock Sufficient

Item ID: 2
Item Name: bat
Current Stock: 60
Reorder Level: 50
Status: Stock Sufficient

Item ID: 3
Item Name: car
Current Stock: 110
Reorder Level: 100
Status: Stock Sufficient

Item ID: 4
Item Name: racket
Current Stock: 100
Reorder Level: 10
Status: Stock Sufficient
PS D:\projects\quest\C>
```

```c
/*
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 ELECTRONICS "Electronics"
#define CLOTHING "Clothing"
#define GROCERIES "Groceries"
#define BOOKS "Books"
struct Discount {
    char category[20];
    float discountPercentage;
    char validTill[15];
};
void updateDiscount(struct Discount *d)
{
    printf("\nUpdating Discount for Category: %s\n", d->category);
    printf("Enter new discount percentage: ");
    scanf("%f", &d->discountPercentage);
    printf("Enter new valid till date (YYYY-MM-DD): ");
    scanf("%s", d->validTill);
}
void displayDiscount(const struct Discount d)
{
    printf("\nCategory: %s\n", d.category);
    printf("Discount Percentage: %.2f%%\n", d.discountPercentage);
    printf("Valid Till: %s\n", d.validTill);
}
void applyDiscount(struct Discount *d)
{
    if (strcmp(d->category, ELECTRONICS) == 0)
```

```
        d->discountPercentage = 10.0;
    else if (strcmp(d->category, CLOTHING) == 0)
        d->discountPercentage = 20.0;
    else if (strcmp(d->category, GROCERIES) == 0)
        d->discountPercentage = 5.0;
    else if (strcmp(d->category, BOOKS) == 0)
        d->discountPercentage = 15.0;
    else
    printf("Invalid category: %s\n", d->category);

}

void main()
{
    int n;
    printf("Enter the number of categories: ");
    scanf("%d", &n);
    struct Discount discounts[n];
    for (int i = 0; i < n; i++)
    {
        printf("\nEnter details for category %d:\n", i + 1);
        printf("Enter category name: ");
        scanf(" %[^\n]s", discounts[i].category);
        applyDiscount(&discounts[i]);
        printf("Enter valid till date (YYYY-MM-DD): ");
        scanf("%s", discounts[i].validTill);
    }
    for (int i = 0; i < n; i++)
        updateDiscount(&discounts[i]);
    printf("\nUpdated Discount Details:\n");
    for (int i = 0; i < n; i++)
        displayDiscount(discounts[i]);
}
```

```
Updating Discount for Category: Electronics
Enter new discount percentage: 10
Enter new valid till date (YYYY-MM-DD): 2025-04-12

Updating Discount for Category: Clothing
Enter new discount percentage: 17
Enter new valid till date (YYYY-MM-DD): 2025-03-25

Updating Discount for Category: Toys
Enter new discount percentage: 2025-06-29
Enter new valid till date (YYYY-MM-DD):
Updated Discount Details:

Category: Electronics
Discount Percentage: 10.00%
Valid Till: 2025-04-12

Category: Clothing
Discount Percentage: 17.00%
Valid Till: 2025-03-25

Category: Toys
Discount Percentage: 2025.00%
Valid Till: -06-29
PS D:\projects\quest\C>
```

```c
/*Create a union that can store an integer, a float, or a character. W
rite a program that assigns values to each member and displays them.*/
#include<stdio.h>
union node{
    int n;
    float f;
    char c;
};
void main()
{
    union node n1,*m;
    m=&n1;
    m->n=6;
    m->f=12.5;
    m->c='a';
```

```
    printf("Integer : %d\n",m->n);
    printf("Float : %.2f\n",m->f);
    printf("Char : %c",m->c);

}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\"
Integer : 1095237729
Float : 12.50
Char : a
PS D:\projects\quest\C>
```

```
/*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 roll;
    char name[10];
};
void main()
{
    union student s;
    printf("Enter roll : ");
    scanf("%d",&s.roll);
    printf("Enter name : ");
    scanf("%s",s.name);
    printf("Name is %s\n",s.name);
    printf("Roll is %d ",s.roll);
}
```

```
Enter roll : 10
Enter name : rahul
Name is rahul
Roll is 1969774962
PS D:\projects\quest\C>
```

```c
/*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 km;
    float miles;
 };
 void main()
 {
    union distance d;
    int n;
    printf("Enter choice(1.km 2.miles)\n");
    scanf("%d",&n);
    if(n==1)
    {
    printf("Enter distance in km :");
    scanf("%f",&d.km);
    printf("Distance in km : %f\n",d.km);
    printf("Distance in miles : %.2f",d.miles*.62137);
    }
    else
    {
        printf("Enter distance in miles : ");
        scanf("%f",&d.miles);
    printf("Distance in km : %f\n",d.miles*1.6093);
    printf("Distance in miles : %.2f",d.miles);
    }


 }
```

```
PS D:\projects\quest\C> cd "d:\projec
Enter choice(1.km 2.miles)
1
Enter distance in km :15
Distance in km : 15.000000
Distance in miles : 9.32
PS D:\projects\quest\C> cd "d:\projec
Enter choice(1.km 2.miles)
2
Enter distance in miles : 10
Distance in km : 16.093000
Distance in miles : 10.00
PS D:\projects\quest\C>
```

```c
/*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;
};
void main()
{
    union employee e;
    int n;
    printf("Enter option(1.id,2.salary) : ");
    scanf("%d",&n);
    if(n==1)
    {
        printf("Enter employee id : ");
        scanf("%d",&e.id);
        printf("Employee id is %d",e.id);
```

```
    }
    else
    {
        printf("Enter employee salary : ");
        scanf("%f",&e.salary);
        printf("Salary is %.2f",e.salary);
    }
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if
Enter option(1.id,2.salary) : 1
Enter employee id : 1001
Employee id is 1001
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if
Enter option(1.id,2.salary) : 2
Enter employee salary : 2500
Salary is 2500.00
PS D:\projects\quest\C>
```

```
/*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 temp;
    float press;
 };
 void main()
 {
    int n;
    union sensor s;
    printf("Enter option (1.temp,2.press) : ");
    scanf("%d",&n);
```

```
    if(n==1)
    {
        printf("Enter the  temperature : ");
        scanf("%f",&s.temp);
        printf("Temperature is %f",s.temp);
    }
    else
    {
        printf("Enter the pressure : ");
        scanf("%f",&s.press);
        printf("Pressure is %f",s.press);
    }
}
```

```
PS D:\projects\quest\C> cd "d:\projects\ques
Enter option (1.temp,2.press) : 1
Enter the  temperature : 100
Temperature is 100.000000
PS D:\projects\quest\C> cd "d:\projects\ques
Enter option (1.temp,2.press) : 2
Enter the pressure : 149
Pressure is 149.000000
PS D:\projects\quest\C>
```

```
/*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 bank
{
    int accno;
    float balance;
};
void main()
{
    union bank b;
    int n;
    printf("Enter option (1.accno,2.balance) : ");
```

```c
    scanf("%d",&n);

    if(n==1)
    {
        printf("Enter the accno : ");
        scanf("%d",&b.accno);
        printf("Accno is %d",b.accno);
    }
    else
    {
        printf("Enter the balance : ");
        scanf("%f",&b.balance);
        printf("Pressure is %f",b.balance);
    }
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest
Enter option (1.accno,2.balance) : 1
Enter the accno : 10001
Accno is 10001
PS D:\projects\quest\C> cd "d:\projects\quest
Enter option (1.accno,2.balance) : 2
Enter the balance : 25000
Pressure is 25000.000000
PS D:\projects\quest\C>
```

```c
/*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 reg;
    float capacity;
};
void main()
```

```c
{
    union vehicle v;
    int n;
    printf("Enter the option(1.reg,2.capacity): ");
    scanf("%d",&n);
    if(n==1)
    {
        printf("Enter the regno : ");
        scanf("%d",&v.reg);
        printf("REG is %d",v.reg);
    }
    else
    {
        printf("Enter the capacity : ");
        scanf("%f",&v.capacity);
        printf("Capacity is %f",v.capacity);
    }
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest
Enter the option(1.reg,2.capacity): 1
Enter the regno : 1001
REG is 1001
PS D:\projects\quest\C> cd "d:\projects\quest
Enter the option(1.reg,2.capacity): 2
Enter the capacity : 400
Capacity is 400.000000
PS D:\projects\quest\C>
```

```c
#include<stdio.h>
union student
{
    int mark;
    char grade;
};
void main()
{
```

```c
    union student s;
    int n;
    printf("Enter the option(1.mark,2.grade): ");
    scanf("%d",&n);
    if(n==1)
    {
        printf("Enter the mark : ");
        scanf("%d",&s.mark);
        printf("Mark is %d",s.mark);
    }
    else
    {
        printf("Enter the grade :");
        getchar();
        scanf("%c",&s.grade);
        printf("Grade is %c",s.grade);
    }
}
```

```
PS D:\projects\quest\C> cd "d:\projects\que
Enter the option(1.mark,2.grade): 1
Enter the mark : 58
Mark is 58
PS D:\projects\quest\C> cd "d:\projects\que
Enter the option(1.mark,2.grade): 2
Enter the grade :C
Grade is C
PS D:\projects\quest\C>
```

```c
/*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>
```

```c
union currency
{
    float eur;
    float usd;
};
void main()
{
    union currency c;
    int n;
    printf("Enter the option(1USD,2.EUR): ");
    scanf("%d",&n);
    if(n==1)
    {
        printf("Enter the USD : ");
        scanf("%f",&c.usd);
        c.eur=c.usd*.97;
        printf("EUR is %f",c.eur);
    }
    else
    {
        printf("Enter the EUR :");
        scanf("%f",&c.eur);
        c.usd=c.eur*1.03;
        printf("USD is %f",c.usd);
    }
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\"
Enter the option(1USD,2.EUR): 1
Enter the USD : 10
EUR is 9.700000
PS D:\projects\quest\C> cd "d:\projects\quest\C\"
Enter the option(1USD,2.EUR): 2
Enter the EUR :10
USD is 10.300000
PS D:\projects\quest\C>
```

```c
/*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>
struct telementry
{
    float temp;
    float vel;
    float alt;
};
union satelite
{
    char img[100];
    struct telementry data;
};
void display(struct telementry t)
{
```

```
    printf("\nTelemetry Data:\n");
    printf("Temperature: %.2f\n", t.temp);
    printf("Velocity: %.2f \n", t.vel);
    printf("Altitude: %.2f \n", t.alt);
}
void image(union satelite *s)
{
    printf("Enter image data :");
    scanf("%s",s->img);
    printf("Processed data : %s\n",s->img);
}
void tel(union satelite *s)
{
    const float temperature = 120.0;
    const float velocity = 30000.0;
    const float altitude = 500.0;

    printf("Enter Telemetry Data:\n");
    printf("Temperature: ");
    scanf("%f", &s->data.temp);
    printf("Velocity : ");
    scanf("%f", &s->data.vel);
    printf("Altitude : ");
    scanf("%f", &s->data.alt);
    if (s->data.temp>temperature)
        printf("Warning: Temperature exceeds safe limit!\n");
    if (s->data.vel>velocity)
        printf("Warning: Velocity exceeds safe limit!\n");
    if (s->data.alt >altitude)
        printf("Warning: Altitude exceeds safe limit!\n");
    display(s->data);
}
void main()
{
    union satelite s;
    int choice;
    do
    {
        printf("1.Process image\n");
        printf("2.Process telementary data\n");
```

```
        printf("3.Exit\n");
        printf("Enter choice\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:image(&s);
            break;
            case 2:tel(&s);
            break;
            case 3:printf("Exiting .....\n");
            break;
            default :printf("Enter valid choice\n");
            break;
        }
    } while (choice!=3);

}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($?) { gcc
1.Process image
2.Process telementary data
3.Exit
Enter choice
1
Enter image data :1234567890
Processed data : 1234567890
1.Process image
2.Process telementary data
3.Exit
Enter choice
2
Enter Telemetry Data:
Temperature: 100
Velocity : 200
Altitude : 400

Telemetry Data:
Temperature: 100.00
Velocity: 200.00
Altitude: 400.00
1.Process image
2.Process telementary data
3.Exit
Enter choice
3
Exiting .....
PS D:\projects\quest\C>
```

```
/*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).
```

```c
/*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>
union Payload {
    char crewDetails[100];
    char cargoDetails[100];
};
struct Mission {
    int missionID;
    char name[50];
    int duration;
    char missionType[10];
    union Payload payload;
};
static int totalMissions = 0;
void addMission(struct Mission *mission);
void updateMission(struct Mission *mission);
void displayMission(struct Mission mission);
int main()
{
    int missionID, found ;
    struct Mission missions[10];
    int choice, i = 0;
    do {
        printf("\nMission Control System:\n");
        printf("1. Add Mission\n");
        printf("2. Update Mission\n");
        printf("3. Display All Missions\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1:
                if (i < 10)
```

```
            {
                addMission(&missions[i]);
                i++;
            }
            else
            printf("Mission storage is full!\n");
            break;

        case 2:
            found = 0;
            printf("Enter Mission ID to update: ");
            scanf("%d", &missionID);

            for (int j = 0; j < i; j++)
            {
                if (missions[j].missionID == missionID)
                {
                    updateMission(&missions[j]);
                    found = 1;
                    break;
                }
            }
            if (!found)
                printf("Mission ID not found.\n");
            break;

        case 3:
            printf("\nMission Summaries:\n");
            for (int j = 0; j < i; j++)
                displayMission(missions[j]);
            break;

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

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

```
    printf("\nTotal Missions Managed: %d\n", totalMissions);
}
void addMission(struct Mission *mission) {
    printf("Enter Mission ID: ");
    scanf("%d", &mission->missionID);
    printf("Enter Mission Name: ");
    scanf(" %[^\n]s", mission->name);
    printf("Enter Mission Duration (in days): ");
    scanf("%d", &mission->duration);
    printf("Enter Mission Type (Crew or Cargo): ");
    scanf("%s", mission->missionType);
    if (strcmp(mission->missionType, "Crew") == 0)
    {
        printf("Enter Crew Details: ");
        scanf(" %[^\n]s", mission->payload.crewDetails);
    }
    else if (strcmp(mission->missionType, "Cargo") == 0)
    {
        printf("Enter Cargo Details: ");
        scanf(" %[^\n]s", mission->payload.cargoDetails);
    }
    else
    {
        printf("Invalid Mission Type. Defaulting to Cargo.\n");
        strcpy(mission->missionType, "Cargo");
        strcpy(mission->payload.cargoDetails, "No cargo details
provided.");
    }
    totalMissions++;
    printf("Mission added successfully.\n");
}
void updateMission(struct Mission *mission) {
    printf("Updating Mission: %s\n", mission->name);
    printf("Enter new duration (in days): ");
    scanf("%d", &mission->duration);
    if (strcmp(mission->missionType, "Crew") == 0)
    {
        printf("Enter new Crew Details: ");
        scanf(" %[^\n]s", mission->payload.crewDetails);
    }
```

```
    else if (strcmp(mission->missionType, "Cargo") == 0)
    {
        printf("Enter new Cargo Details: ");
        scanf(" %[^\n]s", mission->payload.cargoDetails);
    }

    printf("Mission updated successfully.\n");
}
void displayMission(struct Mission mission)
{
    printf("\nMission ID: %d\n", mission.missionID);
    printf("Mission Name: %s\n", mission.name);
    printf("Duration: %d days\n", mission.duration);
    printf("Mission Type: %s\n", mission.missionType);
    if (strcmp(mission.missionType, "Crew") == 0)
     printf("Crew Details: %s\n", mission.payload.crewDetails);
    else if (strcmp(mission.missionType, "Cargo") == 0)
    printf("Cargo Details: %s\n", mission.payload.cargoDetails);
}
```

```
PROBLEMS    OUTPUT    DEBU

Mission updated success

Mission Control System:
1. Add Mission
2. Update Mission
3. Display All Missions
4. Exit
Enter your choice: 33
Invalid choice. Please

Mission Control System:
1. Add Mission
2. Update Mission
3. Display All Missions
4. Exit
Enter your choice: 3

Mission Summaries:

Mission ID: 1
Mission Name: m1
Duration: 45 days
Mission Type: Crew
Crew Details: Operative

Mission ID: 2
Mission Name: m2
Duration: 30 days
Mission Type: Cargo
Cargo Details: Explosiv

Mission Control System:
1. Add Mission
2. Update Mission
3. Display All Missions
4. Exit
Enter your choice: 4
Exiting...
```

```c
/*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>
union MaintenanceType
{
    char routineDetails[100];
    char emergencyDetails[100];
};
struct MaintenanceLog
{
    int logID;
    int aircraftID;
    char date[15];
    char maintenanceType[10];
    union MaintenanceType type;
};
void addLog(struct MaintenanceLog *log);
void displayLogs(struct MaintenanceLog logs[], int count);
int main()
{
    const int MAX_LOGS = 10;
    struct MaintenanceLog logs[MAX_LOGS];
    int choice, logCount = 0;
    do {
        printf("\nAircraft 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);
        switch (choice) {
```

```
            case 1:
                if (logCount < MAX_LOGS)
                 {
                    addLog(&logs[logCount]);
                    logCount++;
                }
                else
                 printf("Log storage is full!\n");

                break;

            case 2:
                displayLogs(logs, logCount);
                break;

            case 3:
                printf("Exiting...\n");
                break;

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

}
void addLog(struct MaintenanceLog *log)
{
    printf("Enter Log ID: ");
    scanf("%d", &log->logID);
    printf("Enter Aircraft ID: ");
    scanf("%d", &log->aircraftID);
    printf("Enter Maintenance Date (DD-MM-YYYY): ");
    scanf("%s", log->date);
    printf("Enter Maintenance Type (Routine or Emergency): ");
    scanf("%s", log->maintenanceType);
    if (strcmp(log->maintenanceType, "Routine") == 0)
    {
        printf("Enter Routine Maintenance Details: ");
        scanf(" %[^\n]s", log->type.routineDetails);
    }
```

```
    else if (strcmp(log->maintenanceType, "Emergency") == 0)
    {
        printf("Enter Emergency Maintenance Details: ");
        scanf(" %[^\n]s", log->type.emergencyDetails);
    }
    else
    {
        printf("Invalid Maintenance Type. Defaulting to Routine.\n");
        strcpy(log->maintenanceType, "Routine");
        strcpy(log->type.routineDetails, "No details provided.");
    }

    printf("Maintenance log added successfully.\n");
}
void displayLogs(struct MaintenanceLog logs[], int count) {
    if (count == 0)
    {
        printf("No maintenance logs available.\n");
        return;
    }
    printf("\nMaintenance Logs:\n");
    for (int i = 0; i < count; i++)
     {
        printf("\nLog ID: %d\n", logs[i].logID);
        printf("Aircraft ID: %d\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("Details: %s\n", logs[i].type.routineDetails);
        else if (strcmp(logs[i].maintenanceType, "Emergency") == 0)
         printf("Details: %s\n", logs[i].type.emergencyDetails);

    }
}
```

```
Maintenance log added successfully.

Aircraft Maintenance Tracker:
1. Add Maintenance Log
2. Display Maintenance Logs
3. Exit
Enter your choice: 1
Enter Log ID: 1002
Enter Aircraft ID: 123
Enter Maintenance Date (DD-MM-YYYY): 2025-01-08
Enter Maintenance Type (Routine or Emergency): E
Enter Emergency Maintenance Details: Fuel leak
Maintenance log added successfully.

Aircraft Maintenance Tracker:
1. Add Maintenance Log
2. Display Maintenance Logs
3. Exit
Enter your choice: 2

Maintenance Logs:

Log ID: 1001
Aircraft ID: 121
Date: 2025-01-10
Maintenance Type: Routine
Details: Engine check

Log ID: 1002
Aircraft ID: 123
Date: 2025-01-08
Maintenance Type: Emergency
Details: Fuel leak

Aircraft Maintenance Tracker:
1. Add Maintenance Log
2. Display Maintenance Logs
3. Exit
Enter your choice: 3
Exiting...
```

```c
/*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>
union NavigationMode
{
    char manualDetails[100];
    char automaticDetails[100];
};
struct NavigationData
{
    float position[3];
    float velocity[3];
    char mode[10];
    union NavigationMode navMode;
};
static int updateCount = 0;
void updateNavigation(struct NavigationData *nav);
void displayNavigation(struct NavigationData nav);
int main()
{
    struct NavigationData spacecraft;
    int choice;
    memset(&spacecraft, 0, sizeof(spacecraft));
    strcpy(spacecraft.mode, "Manual");
    do {
        printf("\nSpacecraft 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);
        switch (choice)
        {
            case 1:
                updateNavigation(&spacecraft);
                break;
            case 2:
                displayNavigation(spacecraft);
                break;
            case 3:
                printf("Exiting...\n");
                break;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    } while (choice != 3);
}
void updateNavigation(struct NavigationData *nav)
{
    printf("Enter Position (x, y, z): ");
    for (int i = 0; i < 3; i++)
    scanf("%f", &nav->position[i]);
    printf("Enter Velocity (vx, vy, vz): ");
    for (int i = 0; i < 3; i++)
    scanf("%f", &nav->velocity[i]);
    printf("Enter Navigation Mode (Manual or Automatic): ");
    scanf("%s", nav->mode);
    if (strcmp(nav->mode, "Manual") == 0)
    {
        printf("Enter Manual Navigation Details: ");
        scanf(" %[^\n]s", nav->navMode.manualDetails);
    }
    else if (strcmp(nav->mode, "Automatic") == 0)
    {
        printf("Enter Automatic Navigation Details: ");
        scanf(" %[^\n]s", nav->navMode.automaticDetails);
    }
    else
    {
        printf("Invalid Navigation Mode. Defaulting to Manual.\n");
```

```
        strcpy(nav->mode, "Manual");
        strcpy(nav->navMode.manualDetails, "No details provided.");
    }
    updateCount++;
    printf("Navigation data updated successfully.\n");
}
void displayNavigation(struct NavigationData nav)
{
    printf("\nNavigation Status:\n");
    printf("Position: (%.2f, %.2f, %.2f)\n", nav.position[0],
nav.position[1], nav.position[2]);
    printf("Velocity: (%.2f, %.2f, %.2f)\n", nav.velocity[0],
nav.velocity[1], nav.velocity[2]);
    printf("Mode: %s\n", nav.mode);
    if (strcmp(nav.mode, "Manual") == 0)
        printf("Details: %s\n", nav.navMode.manualDetails);

    else if (strcmp(nav.mode, "Automatic") == 0)
        printf("Details: %s\n", nav.navMode.automaticDetails);

    printf("Total Updates: %d\n", updateCount);
}
```

```
Enter Velocity (vx, vy, vz): 50 60 70
Enter Navigation Mode (Manual or Automatic): Automatic
Enter Automatic Navigation Details: Free
Navigation data updated successfully.

Spacecraft Navigation System:
1. Update Navigation Data
2. Display Navigation Status
3. Exit
Enter your choice: 1
Enter Position (x, y, z): 33 44 55
Enter Velocity (vx, vy, vz): 10 20 30
Enter Navigation Mode (Manual or Automatic): Manual
Enter Manual Navigation Details:
Forced
Navigation data updated successfully.

Spacecraft Navigation System:
1. Update Navigation Data
2. Display Navigation Status
3. Exit
Enter your choice: 2

Navigation Status:
Position: (33.00, 44.00, 55.00)
Velocity: (10.00, 20.00, 30.00)
Mode: Manual
Details: Forced
Total Updates: 2

Spacecraft Navigation System:
1. Update Navigation Data
2. Display Navigation Status
3. Exit
Enter your choice:
```

```
/*: Create a control system for flight simulations with different aircraft
models.
Requirements:
```

```c
/*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>
union ControlSettings
{
    char manualSettings[100];
    char automatedSettings[100];
};
struct Simulation
{
    int simulationID;
    char aircraftModel[50];
    float duration;
    char controlMode[10];
    union ControlSettings settings;
};
static int totalSimulations = 0;
void startSimulation(struct Simulation *sim);
void updateSettings(struct Simulation *sim);
void displaySimulation(const struct Simulation sim);
int main()
{
    struct Simulation simulations[5];
    int choice;
    int simCount = 0;
    do {
        printf("\nFlight Simulation Control System:\n");
        printf("1. Start a New Simulation\n");
        printf("2. Update Simulation Settings\n");
        printf("3. Display Simulation Results\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
```

```
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                if (simCount < 5)
                {
                    startSimulation(&simulations[simCount]);
                    simCount++;
                }
                else
                printf("Maximum number of simulations reached.\n");
                break;
            case 2:
                if (simCount > 0)
                {
                    int id;
                    printf("Enter Simulation ID to update: ");
                    scanf("%d", &id);
                    int found = 0;
                    for (int i = 0; i < simCount; i++)
                    {
                        if (simulations[i].simulationID == id)
                        {
                            updateSettings(&simulations[i]);
                            found = 1;
                            break;
                        }
                    }
                    if (!found)
                     printf("Simulation ID not found.\n");
                }
                else
                printf("No simulations available to update.\n");
                break;
            case 3:
                if (simCount > 0)
                {
                    for (int i = 0; i < simCount; i++)
                    displaySimulation(simulations[i]);
                }
                else
```

```
                printf("No simulations to display.\n");
                break;
            case 4:
                printf("Exiting...\n");
                break;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    } while (choice != 4);
}
void startSimulation(struct Simulation *sim)
{
    const float MAX_DURATION = 10.0;
    printf("Enter Simulation ID: ");
    scanf("%d", &sim->simulationID);
    printf("Enter Aircraft Model: ");
    scanf(" %[^\n]s", sim->aircraftModel);
    printf("Enter Duration (hours, max %.1f): ", MAX_DURATION);
    scanf("%f", &sim->duration);
    if (sim->duration > MAX_DURATION)
    {
        printf("Duration exceeds maximum limit. Setting to %.1f hours.\n",
MAX_DURATION);
        sim->duration = MAX_DURATION;
    }
    printf("Enter Control Mode (Manual or Automated): ");
    scanf("%s", sim->controlMode);
    if (strcmp(sim->controlMode, "Manual") == 0)
    {
        printf("Enter Manual Control Settings: ");
        scanf(" %[^\n]s", sim->settings.manualSettings);
    }
    else if (strcmp(sim->controlMode, "Automated") == 0)
     {
        printf("Enter Automated Control Settings: ");
        scanf(" %[^\n]s", sim->settings.automatedSettings);
    }
    else
    {
        printf("Invalid control mode. Defaulting to Manual.\n");
```

```
        strcpy(sim->controlMode, "Manual");
        strcpy(sim->settings.manualSettings, "No settings provided.");
    }
    totalSimulations++;
    printf("Simulation started successfully.\n");
}
void updateSettings(struct Simulation *sim)
{
    printf("Updating settings for Simulation ID %d:\n",
sim->simulationID);
    printf("Enter New Control Mode (Manual or Automated): ");
    scanf("%s", sim->controlMode);
    if (strcmp(sim->controlMode, "Manual") == 0)
    {
        printf("Enter New Manual Control Settings: ");
        scanf(" %[^\n]s", sim->settings.manualSettings);
    }
    else if (strcmp(sim->controlMode, "Automated") == 0)
    {
        printf("Enter New Automated Control Settings: ");
        scanf(" %[^\n]s", sim->settings.automatedSettings);
    }
    else
    printf("Invalid control mode. Retaining previous settings.\n");
    printf("Settings updated successfully.\n");
}
void displaySimulation(const struct Simulation sim)
{
    printf("\nSimulation ID: %d\n", sim.simulationID);
    printf("Aircraft Model: %s\n", sim.aircraftModel);
    printf("Duration: %.1f hours\n", sim.duration);
    printf("Control Mode: %s\n", sim.controlMode);
    if (strcmp(sim.controlMode, "Manual") == 0)
        printf("Control Settings: %s\n", sim.settings.manualSettings);
    else if (strcmp(sim.controlMode, "Automated") == 0)
        printf("Control Settings: %s\n", sim.settings.automatedSettings);
    printf("Total Simulations Run: %d\n", totalSimulations);
}
```

```
Enter Duration (hours, max 10.0): 5
Enter Control Mode (Manual or Automated): Automated
Enter Automated Control Settings: Full automatic
Simulation started successfully.

Flight Simulation Control System:
1. Start a New Simulation
2. Update Simulation Settings
3. Display Simulation Results
4. Exit
Enter your choice: 2
Enter Simulation ID to update: 1
Updating settings for Simulation ID 1:
Enter New Control Mode (Manual or Automated): Manual
Enter New Manual Control Settings: Semi manual
Settings updated successfully.

Flight Simulation Control System:
1. Start a New Simulation
2. Update Simulation Settings
3. Display Simulation Results
4. Exit
Enter your choice: 3

Simulation ID: 1
Aircraft Model: m1
Duration: 5.0 hours
Control Mode: Manual
Control Settings: Semi manual
Total Simulations Run: 1

Flight Simulation Control System:
1. Start a New Simulation
2. Update Simulation Settings
3. Display Simulation Results
4. Exit
Enter your choice:
```

```c
/*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.*/
#include <stdio.h>
#include <string.h>
union TestData {
    struct {
        float weight;
        float stress;
    } physicalData;

    char softwareTestResult[100];
};
struct ComponentTest {
    int testID;
    char componentName[50];
    char testType[10];
    union TestData testData;
};
static int totalTests = 0;
void recordTest(struct ComponentTest *test);
void displayTestSummary(const struct ComponentTest test);
int main() {
    struct ComponentTest tests[10];
    int choice, testCount = 0;
    do
    {
        printf("\nAerospace Component Testing System:\n");
        printf("1. Record a New Test\n");
        printf("2. Display Test Summaries\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
```

```
                if (testCount < 10) {
                    recordTest(&tests[testCount]);
                    testCount++;
                }
                else
                printf("Maximum number of tests reached.\n");
                break;
            case 2:
                if (testCount > 0)
                {
                    for (int i = 0; i < testCount; i++)
                        displayTestSummary(tests[i]);
                }
                else
                printf("No tests to display.\n");

                break;

            case 3:
                printf("Exiting...\n");
                break;

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



}
void recordTest(struct ComponentTest *test) {
    printf("Enter Test ID: ");
    scanf("%d", &test->testID);
    printf("Enter Component Name: ");
    scanf(" %[^\n]s", test->componentName);
    printf("Enter Test Type (Physical or Software): ");
    scanf("%s", test->testType);
    if (strcmp(test->testType, "Physical") == 0)
    {
        printf("Enter Weight (kg): ");
        scanf("%f", &test->testData.physicalData.weight);
```

```
        printf("Enter Stress Tolerance (MPa): ");
        scanf("%f", &test->testData.physicalData.stress);
    }
    else if (strcmp(test->testType, "Software") == 0)
    {
        printf("Enter Software Test Results: ");
        scanf(" %[^\n]s", test->testData.softwareTestResult);
    }
    else
    {
        printf("Invalid test type. Defaulting to Physical test with zero
values.\n");
        strcpy(test->testType, "Physical");
        test->testData.physicalData.weight = 0;
        test->testData.physicalData.stress = 0;
    }

    totalTests++;
    printf("Test recorded successfully.\n");
}
void displayTestSummary(const struct ComponentTest test)
{
    printf("\nTest ID: %d\n", test.testID);
    printf("Component Name: %s\n", test.componentName);
    printf("Test Type: %s\n", test.testType);

    if (strcmp(test.testType, "Physical") == 0)
    {
        printf("Weight: %.2f kg\n", test.testData.physicalData.weight);
        printf("Stress Tolerance: %.2f MPa\n",
test.testData.physicalData.stress);
    }
    else if (strcmp(test.testType, "Software") == 0)
    printf("Software Test Results: %s\n",
test.testData.softwareTestResult);


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

```
1. Record a New Test
2. Display Test Summaries
3. Exit
Enter your choice: 1
Enter Test ID: 2
Enter Component Name: case
Enter Test Type (Physical or Software): Physical
Enter Weight (kg): 10
Enter Stress Tolerance (MPa): 9
Test recorded successfully.

Aerospace Component Testing System:
1. Record a New Test
2. Display Test Summaries
3. Exit
Enter your choice: 2

Test ID: 1
Component Name: processor
Test Type: Software
Software Test Results: passed
Total Tests Conducted: 2

Test ID: 2
Component Name: case
Test Type: Physical
Weight: 10.00 kg
Stress Tolerance: 9.00 MPa
Total Tests Conducted: 2

Aerospace Component Testing System:
1. Record a New Test
2. Display Test Summaries
3. Exit
Enter your choice:
```

```c
/*: 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
union RoleDetails
{
    struct
    {
        int yearsOfExperience;
        char expertiseArea[50];
    } engineerDetails;

    struct
    {
        char researchTopic[50];
        int publishedPapers;
    } scientistDetails;
};
struct CrewMember
{
    int crewID;
    char name[50];
    char role[20];
    union RoleDetails roleDetails;
};

static int totalCrew = 0;
void addCrewMember(struct CrewMember *crew);
void updateCrewDetails(struct CrewMember *crew);
void displayCrewDetails(const struct CrewMember crew);
```

```c
int main()
{
    struct CrewMember crewList[MAX_CREW];
    int choice, crewCount = 0;

    do
    {
        printf("\nSpace Station Crew Management System:\n");
        printf("1. Add a New Crew Member\n");
        printf("2. Update Crew Member Details\n");
        printf("3. Display Crew List\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
            case 1:
                if (crewCount < MAX_CREW) {
                    addCrewMember(&crewList[crewCount]);
                    crewCount++;
                }
                else
                printf("Maximum crew members reached.\n");
                break;
            case 2:
                printf("Enter Crew ID to update: ");
                int crewID;
                scanf("%d", &crewID);
                int found = 0;
                for (int i = 0; i < crewCount; i++) {
                    if (crewList[i].crewID == crewID)
                    {
                        updateCrewDetails(&crewList[i]);
                        found = 1;
                        break;
                    }
                }
                if (!found)
```

```
                printf("Crew member with ID %d not found.\n", crewID);
                break;
            case 3:
                if (crewCount > 0) {
                    for (int i = 0; i < crewCount; i++)
                        displayCrewDetails(crewList[i]);
                }
                else
                printf("No crew members available.\n");
                break;
            case 4:
                printf("Exiting...\n");
                break;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    } while (choice != 4);
}
void addCrewMember(struct CrewMember *crew) {
    printf("Enter Crew ID: ");
    scanf("%d", &crew->crewID);
    printf("Enter Crew Name: ");
    scanf(" %[^\n]s", crew->name);
    printf("Enter Role (Engineer/Scientist): ");
    scanf("%s", crew->role);
    if (strcmp(crew->role, "Engineer") == 0)
    {
        printf("Enter Years of Experience: ");
        scanf("%d", &crew->roleDetails.engineerDetails.yearsOfExperience);

        printf("Enter Expertise Area: ");
        scanf(" %[^\n]s",
crew->roleDetails.engineerDetails.expertiseArea);
    }
    else if (strcmp(crew->role, "Scientist") == 0)
    {
        printf("Enter Research Topic: ");
        scanf(" %[^\n]s",
crew->roleDetails.scientistDetails.researchTopic);
```

```
        printf("Enter Number of Published Papers: ");
        scanf("%d", &crew->roleDetails.scientistDetails.publishedPapers);
    }
    else
    {
        printf("Invalid role. Assigning default role as Engineer.\n");
        strcpy(crew->role, "Engineer");
        crew->roleDetails.engineerDetails.yearsOfExperience = 0;
        strcpy(crew->roleDetails.engineerDetails.expertiseArea, "N/A");
    }

    totalCrew++;
    printf("Crew member added successfully.\n");
}
void updateCrewDetails(struct CrewMember *crew)
{
    printf("Updating details for Crew ID %d: %s\n", crew->crewID,
crew->name);
    printf("Enter Role (Engineer/Scientist): ");
    scanf("%s", crew->role);
    if (strcmp(crew->role, "Engineer") == 0)
    {
        printf("Enter Years of Experience: ");
        scanf("%d", &crew->roleDetails.engineerDetails.yearsOfExperience);

        printf("Enter Expertise Area: ");
        scanf(" %[^\n]s",
crew->roleDetails.engineerDetails.expertiseArea);
    }
    else if (strcmp(crew->role, "Scientist") == 0)
    {
        printf("Enter Research Topic: ");
        scanf(" %[^\n]s",
crew->roleDetails.scientistDetails.researchTopic);

        printf("Enter Number of Published Papers: ");
        scanf("%d", &crew->roleDetails.scientistDetails.publishedPapers);
    }
    else
    {
```

```
        printf("Invalid role. Defaulting to Engineer.\n");
        strcpy(crew->role, "Engineer");
        crew->roleDetails.engineerDetails.yearsOfExperience = 0;
        strcpy(crew->roleDetails.engineerDetails.expertiseArea, "N/A");
    }

    printf("Crew member details updated successfully.\n");
}
void displayCrewDetails(const struct CrewMember crew)
{
    printf("\nCrew ID: %d\n", crew.crewID);
    printf("Name: %s\n", crew.name);
    printf("Role: %s\n", crew.role);
    if (strcmp(crew.role, "Engineer") == 0)
    {
        printf("Years of Experience: %d\n",
crew.roleDetails.engineerDetails.yearsOfExperience);
        printf("Expertise Area: %s\n",
crew.roleDetails.engineerDetails.expertiseArea);
    }
    else if (strcmp(crew.role, "Scientist") == 0)
    {
        printf("Research Topic: %s\n",
crew.roleDetails.scientistDetails.researchTopic);
        printf("Published Papers: %d\n",
crew.roleDetails.scientistDetails.publishedPapers);
    }

    printf("Total Crew Members: %d\n", totalCrew);
}
```

```
Space Station Crew Management System:
1. Add a New Crew Member
2. Update Crew Member Details
3. Display Crew List
4. Exit
Enter your choice: 2
Enter Crew ID to update: 1
Updating details for Crew ID 1: c1
Enter Role (Engineer/Scientist): Scientist
Enter Research Topic: Energy
Enter Number of Published Papers: 12
Crew member details updated successfully.

Space Station Crew Management System:
1. Add a New Crew Member
2. Update Crew Member Details
3. Display Crew List
4. Exit
Enter your choice: 3

Crew ID: 1
Name: c1
Role: Scientist
Research Topic: Energy
Published Papers: 12
Total Crew Members: 1

Space Station Crew Management System:
1. Add a New Crew Member
2. Update Crew Member Details
3. Display Crew List
4. Exit
```

```
/*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).
```

```c
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_EXPERIMENTS 10
union Data {
    float numericalData[5];
    char qualitativeData[100];
};
struct ResearchData {
    int experimentID;
    char description[100];
    union Data data;
    int dataType;
};
static int totalAnalyses = 0;
void analyzeData(struct ResearchData *experiment);
void generateReport(const struct ResearchData experiment);
void displayExperimentDetails(const struct ResearchData experiment);
int main()
{
    struct ResearchData experiments[MAX_EXPERIMENTS];
    int choice, experimentCount = 0;

    do
    {
        printf("\nAerospace Research Data Analysis System:\n");
        printf("1. Analyze New Experiment\n");
        printf("2. Display Experiment Report\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
            case 1:
                if (experimentCount < MAX_EXPERIMENTS) {
                    analyzeData(&experiments[experimentCount]);
```

```
                    experimentCount++;
                } else {
                    printf("Maximum experiments reached.\n");
                }
                break;

            case 2:
                printf("Enter Experiment ID to display report: ");
                int experimentID;
                scanf("%d", &experimentID);
                int found = 0;
                for (int i = 0; i < experimentCount; i++)
                {
                    if (experiments[i].experimentID == experimentID)
                    {
                        generateReport(experiments[i]);
                        found = 1;
                        break;
                    }
                }
                if (!found)
                 printf("Experiment with ID %d not found.\n",
experimentID);
                break;

            case 3:
                printf("Exiting...\n");
                break;

            default:
                printf("Invalid choice. Please try again.\n");
        }
    } while (choice != 3);
}
void analyzeData(struct ResearchData *experiment) {
    printf("Enter Experiment ID: ");
    scanf("%d", &experiment->experimentID);
    printf("Enter Experiment Description: ");
    scanf(" %[^\n]s", experiment->description);
    printf("Enter Data Type (0 for numerical, 1 for qualitative): ");
```

```
    scanf("%d", &experiment->dataType);
    if (experiment->dataType == 0)
    {
        printf("Enter 5 numerical data points: ");
        for (int i = 0; i < 5; i++)
        scanf("%f", &experiment->data.numericalData[i]);

    }
    else if (experiment->dataType == 1)
    {
        printf("Enter qualitative data (up to 100 characters): ");
        scanf(" %[^\n]s", experiment->data.qualitativeData);
    }
    else
    {
        printf("Invalid data type.\n");
        return;
    }

    totalAnalyses++;
    printf("Experiment data analyzed successfully.\n");
}
void generateReport(const struct ResearchData experiment)
{
    printf("\n--- Experiment Report ---\n");
    printf("Experiment ID: %d\n", experiment.experimentID);
    printf("Description: %s\n", experiment.description);
    if (experiment.dataType == 0)
    {
        printf("Numerical Data: ");
        for (int i = 0; i < 5; i++)
        printf("%.2f ", experiment.data.numericalData[i]);
        printf("\n");
    }
    else if (experiment.dataType == 1)
        printf("Qualitative Data: %s\n", experiment.data.qualitativeData);


    printf("Total Analyses Conducted: %d\n", totalAnalyses);
}
```

```c
void displayExperimentDetails(const struct ResearchData experiment) {
    printf("\nExperiment ID: %d\n", experiment.experimentID);
    printf("Description: %s\n", experiment.description);
    if (experiment.dataType == 0)
    {
        printf("Numerical Data: ");
        for (int i = 0; i < 5; i++)
        {
            printf("%.2f ", experiment.data.numericalData[i]);
        }
        printf("\n");
    }
    else if (experiment.dataType == 1) {
        printf("Qualitative Data: %s\n", experiment.data.qualitativeData);

}
}
```

```
Aerospace Research Data Analysis System:
1. Analyze New Experiment
2. Display Experiment Report
3. Exit
Enter your choice: 1
Enter Experiment ID: 1
Enter Experiment Description: creating new serum
Enter Data Type (0 for numerical, 1 for qualitative): 0
Enter 5 numerical data points: 7 8 9 11 12
Experiment data analyzed successfully.

Aerospace Research Data Analysis System:
1. Analyze New Experiment
2. Display Experiment Report
3. Exit
Enter your choice: 2
Enter Experiment ID to display report: 1

--- Experiment Report ---
Experiment ID: 1
Description: creating new serum
Numerical Data: 7.00 8.00 9.00 11.00 12.00
Total Analyses Conducted: 1

Aerospace Research Data Analysis System:
1. Analyze New Experiment
2. Display Experiment Report
3. Exit
Enter your choice:
```

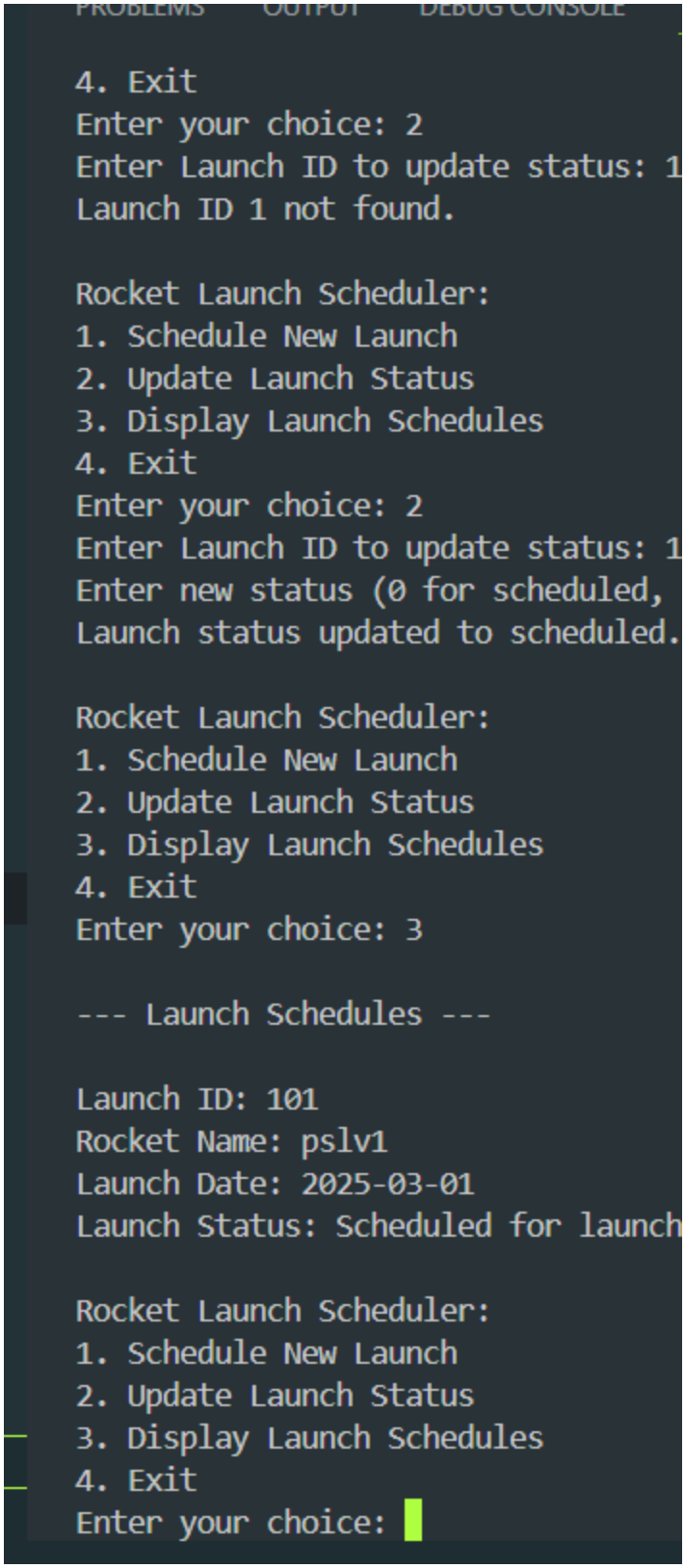
```
/*Create a scheduler for managing rocket launches.
Requirements:
Define a struct for Launch with fields: launchID, rocketName, date, and a
nested union for launch status (scheduled or completed).
Implement functions to schedule launches (call by reference), update
statuses, and display launch schedules (call by value).
```

```c
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 5
union LaunchStatus
{
    char scheduled[50];
    char completed[50];
};
struct Launch
{
    int launchID;
    char rocketName[50];
    char date[20];
    union LaunchStatus status;
    int statusType;
};
void scheduleLaunch(struct Launch *launch);
void updateStatus(struct Launch *launch);
void displayLaunchDetails(struct Launch launch);
int main()
{
    struct Launch launches[MAX_LAUNCHES];
    int choice, launchCount = 0;
    do
    {
        printf("\nRocket Launch Scheduler:\n");
        printf("1. Schedule New 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:
```

```
                if (launchCount < MAX_LAUNCHES)
                {
                    scheduleLaunch(&launches[launchCount]);
                    launchCount++;
                }
                else
                printf("Maximum launches reached.\n");
                break;

            case 2:
                    printf("Enter Launch ID to update status: ");
                    int launchID;
                    scanf("%d", &launchID);
                    int found = 0;
                    for (int i = 0; i < launchCount; i++) {
                        if (launches[i].launchID == launchID)
                        {
                            updateStatus(&launches[i]);
                            found = 1;
                            break;
                        }
                    }
                    if (!found)
                        printf("Launch ID %d not found.\n", launchID);
                break;
            case 3:
                printf("\n--- Launch Schedules ---\n");
                for (int i = 0; i < launchCount; i++)
                    displayLaunchDetails(launches[i]);
                break;
            case 4:
                printf("Exiting...\n");
                break;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    } while (choice != 4);
}
void scheduleLaunch(struct Launch *launch)
{
```

```
    printf("Enter Launch ID: ");
    scanf("%d", &launch->launchID);
    printf("Enter Rocket Name: ");
    scanf(" %[^\n]s", launch->rocketName);
    printf("Enter Launch Date (DD/MM/YYYY): ");
    scanf(" %[^\n]s", launch->date);
    launch->statusType = 0;
    strcpy(launch->status.scheduled, "Scheduled for launch");
    printf("Launch scheduled successfully.\n");
}
void updateStatus(struct Launch *launch)
{
    printf("Enter new status (0 for scheduled, 1 for completed): ");
    scanf("%d", &launch->statusType);
    if (launch->statusType == 0)
    {
        strcpy(launch->status.scheduled, "Scheduled for launch");
        printf("Launch status updated to scheduled.\n");
    }
    else if (launch->statusType == 1)
    {
        strcpy(launch->status.completed, "Completed successfully");
        printf("Launch status updated to completed.\n");
    }
    else
    printf("Invalid status. Please enter 0 or 1.\n");
}
void displayLaunchDetails(struct Launch launch)
{
    printf("\nLaunch ID: %d\n", launch.launchID);
    printf("Rocket Name: %s\n", launch.rocketName);
    printf("Launch Date: %s\n", launch.date);
    if (launch.statusType == 0)
    printf("Launch Status: %s\n", launch.status.scheduled);
    else if (launch.statusType == 1)
    printf("Launch Status: %s\n", launch.status.completed);
}
```


PROBLEMS OUTPUT DEBUG CONSOLE
4. Exit
Enter your choice: 2
Enter Launch ID to update status: 1
Launch ID 1 not found.
Rocket Launch Scheduler:
1. Schedule New Launch
2. Update Launch Status
3. Display Launch Schedules
4. Exit
Enter your choice: 2
Enter Launch ID to update status: 1
Enter new status (0 for scheduled,
Launch status updated to scheduled.
Rocket Launch Scheduler:
1. Schedule New Launch
2. Update Launch Status
3. Display Launch Schedules
4. Exit
Enter your choice: 3
--- Launch Schedules ---
Launch ID: 101
Rocket Name: pslv1
Launch Date: 2025-03-01
Launch Status: Scheduled for launch
Rocket Launch Scheduler:
1. Schedule New Launch
2. Update Launch Status
3. Display Launch Schedules
4. Exit
Enter your choice: