```
1.#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 *totalItems);
void updateItem(Item inventory[], int totalItems);
void displayInventory(Item inventory[], int totalItems);

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

   // Menu loop
   do {
      printf("\n1. Add New Item\n2. Update Item\n3. Display Inventory\n4. Exit\nEnter choice: ");
      scanf("%d", &choice);

      switch (choice) {
         case 1: addItem(inventory, &totalItems); break;
         case 2: updateItem(inventory, totalItems); break;
         case 3: displayInventory(inventory, totalItems); break;
```

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

    return 0;
}
void addItem(Item inventory[], int *totalItems) {
    if (*totalItems < MAX_ITEMS) {
        Item newItem;

        printf("Enter Item ID: ");
        scanf("%d", &newItem.itemID);
        getchar();

        printf("Enter Item Name: ");
        fgets(newItem.itemName, 50, stdin);
        newItem.itemName[strcspn(newItem.itemName, "\n")] = '\0';

        printf("Enter Price: ");
        scanf("%f", &newItem.price);

        printf("Enter Quantity: ");
        scanf("%d", &newItem.quantity);

        inventory[*totalItems] = newItem;
        (*totalItems)++; // Increase item count
        printf("Item added successfully!\n");
    } else {
        printf("Inventory is full!\n");
    }
```

```
}
void updateItem(Item inventory[], int totalItems) {
    int itemID, found = 0;

    printf("Enter Item ID to update: ");
    scanf("%d", &itemID);

    for (int i = 0; i < totalItems; i++) {
        if (inventory[i].itemID == itemID) {
            found = 1;
            printf("Enter new Item Name: ");
            getchar();
            fgets(inventory[i].itemName, 50, stdin);
            inventory[i].itemName[strcspn(inventory[i].itemName, "\n")] = '\0';

            printf("Enter new Price: ");
            scanf("%f", &inventory[i].price);

            printf("Enter new Quantity: ");
            scanf("%d", &inventory[i].quantity);

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

    if (!found) {
        printf("Item with ID %d not found!\n", itemID);
    }
}
void displayInventory(Item inventory[], int totalItems) {
    if (totalItems == 0) {
        printf("No items in inventory.\n");
    } else {
        printf("\nID\tName\t\t\tPrice\tQuantity\n");
        for (int i = 0; i < totalItems; i++) {
            printf("%d\t%-20s\t%.2f\t%d\n", inventory[i].itemID, inventory[i].itemName,
inventory[i].price, inventory[i].quantity);
        }
        printf("Total items: %d\n", totalItems);
    }
}
```

o/p:

```
1. Add New Item
2. Update Item
3. Display Inventory
4. Exit
Enter choice: 1
Enter Item ID: 22
Enter Item Name: Ball
Enter Price: 13
Enter Quantity: 12
Item added successfully!

1. Add New Item
2. Update Item
3. Display Inventory
4. Exit
Enter choice: 1
Enter Item ID: 23
Enter Item Name: bat
Enter Price: 45
Enter Quantity: 10
Item added successfully!

1. Add New Item
2. Update Item
3. Display Inventory
4. Exit
Enter choice: 2
Enter Item ID to update: 22
Enter new Item Name: sketch
Enter new Price: 20
Enter new Quantity: 15
Item updated successfully!

1. Add New Item
2. Update Item
3. Display Inventory
4. Exit
Enter choice: 3

ID      Name                    Price   Quantity
22      sketch          20.00   15
23      bat             45.00   10
Total items: 2

1. Add New Item
2. Update Item
3. Display Inventory
4. Exit
Enter choice: 4
Exiting...
```

```
2.#include <stdio.h>
#include <string.h>
#define MAX_ITEMS 100

#define DISCOUNT_RATE 0.1


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

} Order;


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  *= (1 - DISCOUNT_RATE);


}


void processOrders() {
```

```
int orderCount;
printf("Enter number of orders: ");
scanf("%d", &orderCount);
Order orders[orderCount];

for (int i = 0; i < orderCount; i++) {

  printf("\nEnter Order ID: ");
  scanf("%d", &orders[i].orderID);

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

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

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

  }
  orders[i].totalCost = calculateTotalCost(orders[i]);

  printf("Total Cost before discount: %.2f\n", orders[i].totalCost);

  applyDiscount(&orders[i]);

  printf("Total Cost after discount: %.2f\n", orders[i].totalCost);

}
```

```
}


int main() {
  processOrders();
  return 0;


}
3.#include <stdio.h>
#include <string.h>


typedef struct {
    int customerID;
    char feedbackText[200];
    int rating;  // 1 to 5
} Feedback;


void displayFeedback(Feedback  feedback[], int totalFeedback) {
    printf("\nFeedback  Summary:\n");
    for (int i = 0; i < totalFeedback; i++) {
        printf("\nCustomer  ID: %d\n", feedback[i].customerID);
        printf("Feedback:  %s\n", feedback[i].feedbackText);
        printf("Rating:  %d - ", feedback[i].rating);
        switch (feedback[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() {
    int numFeedback;
    printf("Enter number of feedback entries: ");
    scanf("%d", &numFeedback);

    Feedback feedback[numFeedback];

    for (int i = 0; i < numFeedback; i++) {
        printf("\nEnter feedback details for Customer %d:\n", i + 1);
        printf("Customer ID: ");
        scanf("%d", &feedback[i].customerID);

        printf("Feedback Text: ");
        fgets(feedback[i].feedbackText, sizeof(feedback[i].feedbackText), stdin);
        feedback[i].feedbackText[strcspn(feedback[i].feedbackText, "\n")] = '\0';
        printf("Rating (1-5): ");
        scanf("%d", &feedback[i].rating);
    }
    displayFeedback(feedback, numFeedback);

    return 0;
}
```

o/p:

```
Enter number of feedback entries: 2

Enter feedback details for Customer 1:
Customer ID: 12
Feedback Text: Excellent
Rating (1-5): 4

Enter feedback details for Customer 2:
Customer ID: 13
Feedback Text: Poor
Rating (1-5): 1

Feedback Summary:

Customer ID: 12
Feedback: Excellent
Rating: 4 - Good

Customer ID: 13
Feedback: Poor
Rating: 1 - Very Poor
```

4.
```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAX_ITEMS 100

#define DISCOUNT_RATE 0.1

#define CREDIT_CARD_CHARGE 0.02

#define DEBIT_CARD_CHARGE 0.01

#define PAYPAL_CHARGE 0.03


typedef struct {

    char method[20];

    float amount;

    float transactionCharge;

} Payment;


// Function to convert string to lowercase for case-insensitive comparison

void toLowerCase(char *str) {

    for (int i = 0; str[i]; i++) {

        str[i] = tolower(str[i]);

    }

}


void processPayment(Payment *payment) {


    printf("Enter Payment Method (Credit Card, Debit Card, PayPal): ");

    fgets(payment->method, sizeof(payment->method), stdin);

    payment->method[strcspn(payment->method, "\n")] = '\0'; // Remove the newline character if
any


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


    printf("Enter Payment Amount: ");

    scanf("%f", &payment->amount);
    if (payment->amount < 0) {
        printf("Invalid amount. Amount cannot be negative.\n");
        return;
    }

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

    return 0;
}
```
o/p:
```
Enter Payment Method (Credit Card, Debit Card, PayPal): Credit Card
Enter Payment Amount: 500
Payment Method: Credit Card
Transaction Charge: 10.00
```

5. #include <stdio.h>

```
#include <string.h>

#define MAX_ITEMS 100

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

CartItem cart[MAX_ITEMS];
int totalItems = 0;

void addItem() {
    if (totalItems < MAX_ITEMS) {
        printf("Enter Item ID: ");
        scanf("%d", &cart[totalItems].itemID);
        printf("Enter Item Name: ");
        scanf("%s", cart[totalItems].itemName);
        printf("Enter Price: ");
        scanf("%f", &cart[totalItems].price);
        printf("Enter Quantity: ");
        scanf("%d", &cart[totalItems].quantity);
        totalItems++;
        printf("Item added successfully!\n");
    } else {
        printf("Cart is full!\n");
    }
}
```

```c
void removeItem(int itemID) {
    for (int i = 0; i < totalItems; i++) {
        if (cart[i].itemID == itemID) {
            for (int j = i; j < totalItems - 1; j++) {
                cart[j] = cart[j + 1];
            }
            totalItems--;
            printf("Item removed successfully!\n");
            return;
        }
    }
    printf("Item not found!\n");
}

void displayCart() {
    printf("\n--- Shopping Cart ---\n");
    for (int i = 0; i < totalItems; i++) {
        printf("Item 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);
        printf("--------------------\n");
    }
    printf("Total Items: %d\n", totalItems);
}

int main() {
    int choice, itemID;
    do {
        printf("\nShopping Cart System\n");
        printf("1. Add Item\n");
        printf("2. Remove Item\n");
        printf("3. Display Cart\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                addItem();
                break;
            case 2:
                printf("Enter Item ID to remove: ");
                scanf("%d", &itemID);
                removeItem(itemID);
                break;
            case 3:
                displayCart();
                break;
            case 4:
                printf("Exiting the system.\n");
                break;
            default:
                printf("Invalid choice. Try again.\n");

        }

    } while (choice != 4);
    return 0;

}
```

o/p:

```
Shopping  Cart System

1. Add Item

2. Remove Item

3. Display Cart

4. Exit

Enter your choice: 1

Enter Item ID: 101

Enter Item Name: Laptop

Enter Price: 800.00

Enter Quantity: 2

Item added successfully!Shopping  Cart System

1. Add Item

2. Remove Item

3. Display Cart

4. Exit

Enter your choice: 1

Enter Item ID: 102

Enter Item Name: Smartphone

Enter Price: 500.00

Enter Quantity: 1

Item added successfully!


Shopping  Cart System

1. Add Item

2. Remove Item

3. Display Cart

4. Exit

Enter your choice: 3


--- Shopping  Cart ---
```

```
Item ID: 101
Item Name: Laptop
Price: 800.00
Quantity: 2
--------------------
Item ID: 102
Item Name: Smartphone
Price: 500.00
Quantity: 1
--------------------
Total Items: 2
 Shopping  Cart System
1. Add Item
2. Remove Item
3. Display Cart
4. Exit
Enter your choice: 2
Enter Item ID to remove: 101
Item removed successfully!
Shopping  Cart System
1. Add Item
2. Remove Item
3. Display Cart
4. Exit
Enter your choice: 3
--- Shopping  Cart ---
Item ID: 102
Item Name: Smartphone
Price: 500.00
Quantity:
Total Items: 1
```

6.
```
#include <stdio.h>
#include <string.h>

#define MAX_PRODUCTS 5

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

int searchByName(struct Product products[], int count, const char *name) {
    for (int i = 0; i < count; i++) {
        if (strcmp(products[i].productName, name) == 0) {
            return i;
        }
    }
    return -1;
}

void updateProduct(struct Product *product) {
    printf("Enter new product name: ");
    getchar();
    fgets(product->productName, sizeof(product->productName), stdin);
    product->productName[strcspn(product->productName, "\n")] = 0;

    printf("Enter new category: ");
    fgets(product->category, sizeof(product->category), stdin);
    product->category[strcspn(product->category, "\n")] = 0;
    printf("Enter new price: ");
    scanf("%f", &product->price);
}


void displayProduct(struct Product product) {
    printf("Product ID: %d\n", product.productID);
    printf("Product Name: %s\n", product.productName);
    printf("Category: %s\n", product.category);
    printf("Price: %.2f\n", product.price);
}

int main() {
    struct Product products[MAX_PRODUCTS] = {
        {101, "Laptop", "Electronics", 800.50},
        {102, "Phone", "Electronics", 500.30},
        {103, "Shampoo", "Personal Care", 15.75},
        {104, "Watch", "Accessories", 120.00},
        {105, "Table", "Furniture", 150.99}
    };

    int choice;
    char searchName[50];

    printf("Product Search System\n");
    printf("1. Search by Name\n");
    printf("2. Update Product Details\n");
    printf("3. Exit\n");

    while(1) {
        printf("\nEnter your choice: ");
```

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

if (choice == 1) {
    printf("Enter product name to search: ");
    getchar();
    fgets(searchName, sizeof(searchName), stdin);
    searchName[strcspn(searchName, "\n")] = 0;

    int index = searchByName(products, MAX_PRODUCTS, searchName);
    if (index != -1) {
        printf("\nProduct found:\n");
        displayProduct(products[index]);
    } else {
        printf("Product not found!\n");
    }
}
else if (choice == 2) {
    int id;
    printf("Enter the Product ID to update: ");
    scanf("%d", &id);

    int index = -1;
    for (int i = 0; i < MAX_PRODUCTS; i++) {
        if (products[i].productID == id) {
            index = i;
            break;
        }
    }

    if (index != -1) {
        printf("\nUpdating product with ID %d:\n", products[index].productID);
```

```
            updateProduct(&products[index]);
            printf("Product details updated successfully!\n");
        } else {
            printf("Product with ID %d not found!\n", id);
        }
    }
    else if (choice == 3) {
        printf("Exiting the system.\n");
        break;
    }
    else {
        printf("Invalid choice. Please try again.\n");
    }
  }

  return 0;
}
```

o/p:

```
Product Search System
1. Search by Name
2. Update Product Details
3. Exit

Enter your choice: 1
Enter product name to search: Laptop
Product found:
Product ID: 101
Product Name: Laptop
Category: Electronics
Price:800.50
```

8. #include <stdio.h>

```
#include <string.h>

#define REWARD_RATE 0.05



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

void updateRewardPoints(Customer *customer) {
    customer->rewardPoints = customer->totalPurchases * REWARD_RATE;
}

int main() {
    Customer customer;

    // Input customer details
    printf("Enter Customer ID: ");
    scanf("%d", &customer.customerID);
    getchar();

    printf("Enter Customer Name: ");
    fgets(customer.name, sizeof(customer.name), stdin);
    customer.name[strcspn(customer.name, "\n")] = '\0';
    printf("Enter Total Purchases: ");
    scanf("%f", &customer.totalPurchases);
    updateRewardPoints(&customer);



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

    return 0;
}
```

o/p:

```
Enter Customer ID: 1
Enter Customer Name: Sofia
Enter Total Purchases: 100

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

10.

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

#define ELECTRONICS_DISCOUNT 0.10  // 10% discount for Electronics
#define FURNITURE_DISCOUNT 0.15    // 15% discount for Furniture
#define CLOTHING_DISCOUNT 0.20     // 20% discount for Clothing

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

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

int main() {
    Discount discount;

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

    applyDiscount(&discount);

    printf("\nDiscount Details:\n");
    printf("Category: %s\n", discount.category);
    printf("Discount Percentage: %.2f%%\n", discount.discountPercentage * 100);
    return 0;
}
```

o/p:

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

Discount Details:
Category: Electronics
Discount Percentage: 10.00%
```

1.

```c
#include <stdio.h>

union DataTypes {
    int i;
    float f;
    char c;
};

int main() {
    union DataTypes data;
    data.i = 10;
    printf("Integer value: %d\n", data.i);
    data.f = 3.14f;
    printf("Float value: %.2f\n", data.f);
    data.c = 'A';
    printf("Character value: %c\n", data.c);

    return 0;
}
```

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

union Student {
    int rollno;
    char name[20];
};

int main() {
    union Student students;
    int choice;

    printf("1. Roll Number\n2. Name\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

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

    return 0;
}
```

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

union Distance {
    float kilometers;
    float miles;
};

int main() {
    union Distance dist;
    int choice;

    printf("Choose the unit of distance:\n");
    printf("1. Kilometers\n2. Miles\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    if (choice == 1) {
        printf("Enter distance in kilometers: ");
        scanf("%f", &dist.kilometers);

        float miles = dist.kilometers * 0.621371;
        printf("Distance in kilometers: %.2f\n", dist.kilometers);
        printf("Converted distance in miles: %.2f\n", miles);
    }
    else if (choice == 2) {
        printf("Enter distance in miles: ");
        scanf("%f", &dist.miles);

        float kilometers = dist.miles * 1.60934;
        printf("Distance in miles: %.2f\n", dist.miles);
        printf("Converted distance in kilometers: %.2f\n", kilometers);
    }
    else {
        printf("Invalid choice!\n");
    }

    return 0;
}

4.#include <stdio.h>
#include <math.h>

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

int main() {
    union Shape shape;
    int choice;

    printf("Select a shape:\n");
    printf("1. Circle\n");
    printf("2. Rectangle\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    if (choice == 1) {
        printf("Enter  radius of the circle: ");
        scanf("%f",  &shape.radius);



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

        float area = shape.length  * shape.width;
        printf("Area  of the rectangle: %.2f\n", area);
    }
    else {
        printf("Invalid  choice.\n");
    }

    return 0;
}

5.#include <stdio.h>

union  Employee {
    int id;
    float salary;
};

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

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

    if (choice == 1) {
        printf("Enter employee ID: ");
        scanf("%d", &employee.id);

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

        printf("Enter employee salary: ");
        scanf("%f", &employee.salary);

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

    return 0;
}
```

6.# include <stdio.h>

```
union SensorData{
    float temp;
    float p;
};
int main(){
    union SensorData  data;
    int choice;

    printf("Select  ur Readings:\n");
    printf("  1.Temperature\n  2.Pressure\n");
    printf("Enter  Your  Choice:");
    scanf("%d",&choice);

    if(choice==1){
        printf("Enter  the temp:");
        scanf("%f",&data.temp);

        printf("Data  temperature:%f\n",data.temp);
    }
    else if(choice==2){
        printf("Enter  the Pressure:");
        scanf("%f",&data.p);

        printf("Data  Pressure:%f\n",data.p);
    }
    else{
        printf("Invalid");
    }
}
```

```
7.#include <stdio.h>
union  BankAccount{
   int num;
   float balance;
};

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

   printf(" 1.Account  Number\n  2.Balance\n");
   printf("Enter  your  choice:");
   scanf("%d",&choice);

   if(choice==1){
      printf("Enter  the acc num:");
      scanf("%d",&Account.num);

      printf("Account  Number:%d\n",Account.num);
   }
   else if(choice==2){
      printf("Enter  the Balance:");
      scanf("%2f",&Account.balance);

      printf("Account  Balance:%2f\n",Account.balance);
   }
   else{
      printf("invalid");
   }
   return 0;
}
```

```
8.# include <stdio.h>
union Vehicle{
   int regnum;
   float fuelCap;
};

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

   printf(" 1.Vehicle Reg Num\n 2.Fuel capacity\n");
   printf("Enter your choice:");
   scanf("%d",&choice);

   if (choice==1){
      printf("Enter the regnum:");
      scanf("%d",&vehicle.regnum);

      printf("Vehicle reg Num:%d\n",vehicle.regnum);
   }
   else if(choice==2){
      printf("Enter the fuel cap:");
      scanf("%f",&vehicle.fuelCap);

      printf("Vehicle reg Num:%f\n",vehicle.fuelCap);
   }
   else{
      printf("Invalid");
   }
}
```

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

union Student {
    int marks;
    char grade;
};

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

    printf(" 1. Marks\n 2. Grade\n");
    printf("Enter your choice (1 or 2): ");
    scanf("%d", &choice);

    if (choice == 1) {
        printf("Enter student's marks: ");
        scanf("%d", &student.marks);
        printf("Student's Marks: %d\n", student.marks);
    }
    else if (choice == 2) {
        printf("Enter student's grade: ");
        scanf(" %c", &student.grade);
        printf("Student's Grade: %c\n", student.grade);
    }
    else {
        printf("Invalid choice.\n");
    }

    return 0;
}
```

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

union Currency {
   float usd;
   float eur;
};

int main() {
   union Currency currency;
   int choice;

   printf("1. USD\n");
   printf("2. EUR\n");
   printf("Enter your choice (1 or 2): ");
   scanf("%d", &choice);

   if (choice == 1) {
      printf("Enter the amount in USD: ");
      scanf("%f", &currency.usd);
      printf("Equivalent in EUR: %.2f\n", currency.usd * 0.85);
   }
   else if (choice == 2) {
      printf("Enter the amount in EUR: ");
      scanf("%f", &currency.eur);
      printf("Equivalent in USD: %.2f\n", currency.eur * 1.18);
   }
   else {
      printf("Invalid choice.\n");
   }
```

```
    return 0;
}
```

Problem 1: Aircraft Fleet Management

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

Requirements:

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

Output Expectations:

- Display updated fleet information after each operation.

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

#define MAX_FLEET_SIZE 100

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

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

// Function to add new aircraft
```

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

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

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

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

int main() {
    Aircraft fleet[MAX_FLEET_SIZE];

    // Add new aircraft to the fleet
    addAircraft(&fleet[0], 101, "Boeing 737", 200, "Available");
    addAircraft(&fleet[1], 102, "Airbus A320", 180, "In Service");

    // Display fleet details
    displayFleet(fleet);

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

    // Display updated fleet details
    displayFleet(fleet);

    return 0;
}
```

O/p:

```
Fleet Details:
```

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

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

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

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

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

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

Problem 2: Satellite Data Processing

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

Requirements:

- Define a union for SatelliteData to store either image data (array) or telemetry data (nested structure).

- Use struct to define Telemetry with fields: temperature, velocity, and altitude.
- Implement functions to process image and telemetry data (call by reference).
- Use const for fixed telemetry limits.
- Employ loops to iterate through data points.

Output Expectations:

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

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

#define MAX_IMAGE_SIZE 5
#define MAX_TELEMETRY_DATA 3

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

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

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

```
}

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

int main() {
    SatelliteData data;
    int choice;

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

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

    return 0;
}
```

```
O/p: Select the type of satellite data:
1. Image Data
2. Telemetry Data
Enter your choice (1 or 2): 1
Enter pixel 1 value: 150
Enter pixel 2 value: 300
Enter pixel 3 value: 200
Enter pixel 4 value: 234
Enter pixel 5 value: 120
Processing Image Data:
Pixel 1: 150
Pixel 2: 300
Pixel 3: 200
Pixel 4: 234
Pixel 5: 120
Select the type of satellite data:
1. Image Data
2. Telemetry Data
Enter your choice (1 or 2): 2
Enter temperature (°C): 21

Enter velocity (km/s): 45

Enter altitude (km): 546

Processing Telemetry Data:

Temperature: 21.00°C

Velocity: 45.00 km/s

Altitude: 546.00 km
```

Problem 3: Mission Control System

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

Requirements:

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

Output Expectations:

- Provide detailed mission summaries including payload information.

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

#include <string.h>


#define MAX_MISSIONS 5


// Struct for Crew

typedef struct {

   char name[50];

   int age;

   char role[50];

} Crew;


// Struct for Cargo

typedef struct {
    char description[50];
    float weight;
} Cargo;

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

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

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

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

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

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

int main() {
    Mission missions[MAX_MISSIONS];

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

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

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

O/p:

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

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

Problem 4: Aircraft Maintenance Tracker

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

Requirements:

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

Output Expectations:

- Display maintenance logs categorized by type.

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

#define MAX_LOGS 10
#define MAINTENANCE_FREQUENCY 30 // Days between routine maintenance
```

```
// Structs for different types of maintenance
typedef struct {
   char description[100];
} RoutineMaintenance;

typedef struct {
   char issue[100];
   char action[100];
} EmergencyMaintenance;

// Union to store either Routine or Emergency maintenance
typedef union {
   RoutineMaintenance routine;
   EmergencyMaintenance emergency;
} MaintenanceType;

// Struct for Maintenance Log
typedef struct {
   int logID;
   int aircraftID;
   char date[20];
   MaintenanceType maintenance;
   int isEmergency;  // 1 for emergency, 0 for routine
} MaintenanceLog;

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

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

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

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

int main() {
   MaintenanceLog logs[MAX_LOGS];

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

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

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

   return 0;
}
```

O/p:

```
Log ID: 1
Aircraft ID: 101
Date: 2025-01-10
Maintenance Type: Routine
Description: Routine inspection and oil change

Log ID: 2
Aircraft ID: 102
Date: 2025-01-12
Maintenance Type: Emergency
Issue: Engine failure
Action: Replaced faulty engine
```

Problem 5: Spacecraft Navigation System

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

Requirements:

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

- Use loops and switch case for managing navigation modes.

Output Expectations:

- Show updated position and velocity with navigation mode details.

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

#define MAX_UPDATES 5

// Struct for manual mode navigation data
typedef struct {
    float x, y, z;  // Position coordinates
} ManualNavigation;

// Struct for automatic mode navigation data
typedef struct {
    float x, y, z;  // Position coordinates
    float velocity; // Velocity
} AutomaticNavigation;

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

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

```
// Static variable to count navigation updates

static int totalUpdates = 0;


// Function to update navigation data

void updateNavigationData(NavigationData* data, int updateID, const char* mode, float x, float y,
float z, float velocity) {

    data->updateID = updateID;

    strcpy(data->mode, mode);


    if (strcmp(mode, "Manual") == 0) {

        data->navMode.manual.x = x;

        data->navMode.manual.y = y;

        data->navMode.manual.z = z;

    } else if (strcmp(mode, "Automatic") == 0) {

        data->navMode.automatic.x = x;

        data->navMode.automatic.y = y;

        data->navMode.automatic.z = z;

        data->navMode.automatic.velocity = velocity;

    }


    totalUpdates++;

}


// Function to display navigation data

void displayNavigationData(NavigationData data) {

    printf("\nUpdate ID: %d\n", data.updateID);

    printf("Navigation Mode: %s\n", data.mode);


    if (strcmp(data.mode, "Manual") == 0) {

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

int main() {
    NavigationData updates[MAX_UPDATES];

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

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

    return 0;
}
```

O/p:

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

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

Problem 6: Flight Simulation Control

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

Requirements:

- Define a struct for Simulation with fields: simulationID, aircraftModel, duration, and a nested union for control settings (manual or automated).
- Implement functions to start simulations (call by reference), update settings, and display simulation results (call by value).
- Use const for fixed simulation parameters.
- Utilize loops to run multiple simulations and a switch case for selecting control settings.

Output Expectations:

- Display simulation results with control settings.

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

#define MAX_SIMULATIONS 3

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

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

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

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

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

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

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

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

int main() {
   Simulation simulations[MAX_SIMULATIONS];

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

   // Display simulation results
   for (int i = 0; i < MAX_SIMULATIONS; i++) {
      displaySimulationResults(simulations[i]);
   }

   return 0;
}
```

Op:

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

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

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

Problem 7: Aerospace Component Testing

Description: Develop a system for testing different aerospace components.

Requirements:

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

Output Expectations:

- Display test results categorized by component type.

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

#define MAX_TESTS 3

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

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

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

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

```
// Static variable to track total tests

static int totalTests = 0;


// Function to record test results

void recordTestResult(ComponentTest *test, int testID, const char *component, const char *type,
float param1, float param2, const char *status) {

    test->testID = testID;

    strcpy(test->componentName, component);

    strcpy(test->testType, type);


    if (strcmp(type, "Physical") == 0) {

        test->data.physical.temperature = param1;

        test->data.physical.pressure = param2;

    } else if (strcmp(type, "Software") == 0) {

        test->data.software.errorCode = (int)param1;

        strcpy(test->data.software.status, status);

    }


    totalTests++;

}


// Function to display test result

void displayTestResult(const ComponentTest test) {

    printf("\nTest ID: %d\n", test.testID);

    printf("Component: %s\n", test.componentName);

    printf("Test Type: %s\n", test.testType);


    if (strcmp(test.testType, "Physical") == 0) {

        printf("Physical Test - Temperature: %.2f, Pressure: %.2f\n", test.data.physical.temperature,
test.data.physical.pressure);

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

int main() {
    ComponentTest tests[MAX_TESTS];

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

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

    return 0;
}
```

O/p:

```
Test ID: 1
Component: Engine
Test Type: Physical
Physical Test - Temperature: 100.50, Pressure: 15.20

Test ID: 2
Component: Flight Software
Test Type: Software
Software Test - Error Code: 0, Status: Passed
```

Test ID: 3

Component: Wing Structure

Test Type: Physical

Physical Test - Temperature: 80.00, Pressure: 12.00

Problem 8: Space Station Crew Management

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

Requirements:

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

Output Expectations:

- Show updated crew information including role-specific details.

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

#include <string.h>


#define MAX_CREW 3


// Struct for engineer role-specific details

typedef struct {

   char specialty[50];

   int yearsExperience;

} Engineer;


// Struct for scientist role-specific details

typedef struct {

   char field[50];

   int publications;
} Scientist;


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


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


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


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

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

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

int main() {
   CrewMember crew[MAX_CREW];

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

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

   return 0;
}
```

O/p:

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

Crew ID: 2
Name: Bob
Role: Scientist
Field: Astrophysics
Publications: 5

Crew ID: 3
Name: Charlie
Role: Engineer
Specialty: Robotics
Years of Experience: 8
```

Problem 9: Aerospace Research Data Analysis

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

Requirements:

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

Output Expectations:

- Provide detailed reports of analyzed data.

Sol: #include <stdio.h>

```
#include <string.h>

#define MAX_ANALYSES 3

// Struct for numerical data
typedef struct {
    float value1;
    float value2;
} NumericalData;

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

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

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

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

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

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

    totalAnalyses++;
}

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

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

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

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

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

   return 0;
}
```

O/p:

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

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

Experiment ID: 103
Description: Temperature Test
Data Type: Numerical
```

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

Problem 10: Rocket Launch Scheduler

Description: Create a scheduler for managing rocket launches.

Requirements:

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

Output Expectations:

- Display detailed launch schedules and statuses.

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

#define MAX_LAUNCHES 5

// Struct for launch date
typedef struct {
    int year, month, day;  // Launch date
} LaunchDate;

// Union for launch status (either scheduled or completed)
typedef union {
    char scheduled[20];  // "Scheduled"
    char completed[20];  // "Completed"
} LaunchStatus;

// Struct for launch details
typedef struct {
    int launchID;

    char rocketName[50];

    LaunchDate date;

    LaunchStatus status;  // Union to hold launch status

    char statusType[20];  // "Scheduled" or "Completed"

} Launch;


// Function to schedule a launch

void scheduleLaunch(Launch* launch, int launchID, const char* rocketName, int year, int month, int
day, const char* status) {

    launch->launchID = launchID;

    strcpy(launch->rocketName, rocketName);

    launch->date.year = year;

    launch->date.month = month;

    launch->date.day = day;

    strcpy(launch->statusType, status);


    if (strcmp(status, "Scheduled") == 0) {

        strcpy(launch->status.scheduled, "Scheduled");

    } else if (strcmp(status, "Completed") == 0) {

        strcpy(launch->status.completed, "Completed");

    }

}


// Function to update the launch status

void updateLaunchStatus(Launch* launch, const char* status) {

    strcpy(launch->statusType, status);

    if (strcmp(status, "Scheduled") == 0) {

        strcpy(launch->status.scheduled, "Scheduled");

    } else if (strcmp(status, "Completed") == 0) {

        strcpy(launch->status.completed, "Completed");
    }
}

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

int main() {
    Launch launches[MAX_LAUNCHES];

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

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

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

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

    return 0;
}
```

O/p:

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

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

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

Updated Launch Schedules:

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

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

Launch ID: 3

Rocket Name: Atlas V

Launch Date: 2025-7-10

Launch Status: Scheduled