1. **Student Information**:
   - Define a structure to store student information, including name, roll number, and marks in three subjects.
   - Write a program to input data for 5 students and display the details along with their average marks.
2. **Employee Details**:
   - Create a structure to store employee details like name, ID, salary, and department.
   - Write a function to display the details of employees whose salary is above a certain threshold.
3. **Book Store Inventory**:
   - Define a structure to represent a book with fields for title, author, ISBN, and price.
   - Write a program to manage an inventory of books and allow searching by title.
4. **Date Validation**:
   - Create a structure to represent a date with day, month, and year.
   - Write a function to validate if a given date is correct (consider leap years).
5. **Complex Numbers**:
   - Define a structure to represent a complex number with real and imaginary parts.
   - Implement functions to add, subtract, and multiply two complex numbers.
6. **Bank Account**:
   - Design a structure to store information about a bank account, including account number, account holder name, and balance.
   - Write a function to deposit and withdraw money, and display the updated balance.
7. **Car Inventory System**:
   - Create a structure for a car with fields like make, model, year, and price.
   - Write a program to store details of multiple cars and print cars within a specified price range.
8. **Library Management**:
   - Define a structure for a library book with fields for title, author, publication year, and status (issued or available).
   - Write a function to issue and return books based on their status.
9. **Student Grades**:
   - Create a structure to store a student's name, roll number, and an array of grades.
   - Write a program to calculate and display the highest, lowest, and average grade for each student.
10. **Product Catalog**:
    - Define a structure to represent a product with fields for product ID, name, quantity, and price.
    - Write a program to update the quantity of products after a sale and calculate the total sales value.

```
#include <stdio.h>


// Define structure to store student information

struct Student {

    char name[50];

    int rollNumber;

    float marks[3];

    float average;

};


// Function to calculate average marks

float calculateAverage(float marks[], int size) {

    float sum = 0.0;

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

        sum += marks[i];

    }

    return sum / size;

}


int main() {

    struct Student students[5];

    int i, j;
    // Input student data
    for (i = 0; i < 5; i++) {
        printf("\nEnter details for student %d:\n", i + 1);
        printf("Name: ");
        scanf("%s", students[i].name);
        printf("Roll Number: ");
        scanf("%d", &students[i].rollNumber);
        printf("Enter marks for 3 subjects: ");
        for (j = 0; j < 3; j++) {
            scanf("%f", &students[i].marks[j]);
        }
        // Calculate average
        students[i].average = calculateAverage(students[i].marks, 3);
    }

    // Display student details
    printf("\nStudent Details:\n");
    for (i = 0; i < 5; i++) {
        printf("\nName: %s", students[i].name);
        printf("\nRoll Number: %d", students[i].rollNumber);
        printf("\nMarks: %.2f, %.2f, %.2f", students[i].marks[0], students[i].marks[1],
students[i].marks[2]);
        printf("\nAverage Marks: %.2f\n", students[i].average);
    }
    return 0;
}
```

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

// Define structure to store employee details
struct Employee {
    char name[50];
    int id;
    float salary;
    char department[50];
};

// Function to display employees with salary above a threshold
void displayHighSalaryEmployees(struct Employee employees[], int size, float threshold) {
    printf("\nEmployees with salary above %.2f:\n", threshold);
    for (int i = 0; i < size; i++) {
        if (employees[i].salary > threshold) {
            printf("\nName: %s", employees[i].name);
            printf("\nID: %d", employees[i].id);
            printf("\nSalary: %.2f", employees[i].salary);
            printf("\nDepartment: %s\n", employees[i].department);
        }
    }
}

int main() {
    struct Employee employees[3];

    // Input employee details
    for (int i = 0; i < 3; i++) {
        printf("\nEnter details for employee %d:\n", i + 1);
        printf("Name: ");
        scanf("%s", employees[i].name);
        printf("ID: ");
        scanf("%d", &employees[i].id);
        printf("Salary: ");
        scanf("%f", &employees[i].salary);
        printf("Department: ");
        scanf("%s", employees[i].department);
    }

    // Display employees with salary above a threshold
    float threshold;
    printf("\nEnter salary threshold: ");
    scanf("%f", &threshold);
    displayHighSalaryEmployees(employees, 3, threshold);return 0;}
```

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

// Define structure to represent a book
struct Book {
    char title[100];
    char author[100];
    char ISBN[20];
    float price;
};

// Function to search for a book by title
void searchBookByTitle(struct Book books[], int size, char title[]) {
    int found = 0;
    for (int i = 0; i < size; i++) {
        if (strcmp(books[i].title, title) == 0) {
            printf("\nBook found:\n");
            printf("Title: %s\n", books[i].title);
            printf("Author: %s\n", books[i].author);
            printf("ISBN: %s\n", books[i].ISBN);
            printf("Price: %.2f\n", books[i].price);
            found = 1;
            break;
        }
    }
    if (!found) {
        printf("\nBook not found.\n");
    }
}
```

```
int main() {
    struct Book inventory[3];

    // Input book details
    for (int i = 0; i < 3; i++) {
        printf("\nEnter details for book %d:\n", i + 1);
        printf("Title: ");
        scanf(" %[^"]", inventory[i].title);
        printf("Author: ");
        scanf(" %[^"]", inventory[i].author);
        printf("ISBN: ");
        scanf("%s", inventory[i].ISBN);
        printf("Price: ");
        scanf("%f", &inventory[i].price);
    }

    // Search for a book by title
    char searchTitle[100];
    printf("\nEnter the title of the book to search: ");
    scanf(" %[^"]", searchTitle);
    searchBookByTitle(inventory, 3, searchTitle);

    return 0;
}
```

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

// Define structure to represent a date
struct Date {
    int day;
    int month;
    int year;
};

// Function to check for a leap year
bool isLeapYear(int year) {
    if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {
        return true;
    }
    return false;
}

// Function to validate the date
bool validateDate(struct Date date) {
    int daysInMonth[] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

    // Check for leap year and adjust February days
    if (isLeapYear(date.year)) {
        daysInMonth[2] = 29;
    }

    // Validate month and day
    if (date.month < 1 || date.month > 12) {
        return false;
    }
    if (date.day < 1 || date.day > daysInMonth[date.month]) {
        return false;
    }
    return true;
}

int main() {
    struct Date date;

    // Input date from user
    printf("Enter date (DD MM YYYY): ");
    scanf("%d %d %d", &date.day, &date.month, &date.year);

    // Validate the date
    if (validateDate(date)) {
        printf("The date is valid.\n");
    } else {
        printf("The date is invalid.\n");
    }

    return 0;
}
```

```
#include <stdio.h>

// Define structure to represent a complex number
struct Complex {
   float real;
   float imaginary;
};

// Function to add two complex numbers
struct Complex add(struct Complex a, struct Complex b) {
   struct Complex result;
   result.real = a.real + b.real;
   result.imaginary = a.imaginary + b.imaginary;
   return result;
}

// Function to subtract two complex numbers
struct Complex subtract(struct Complex a, struct Complex b) {
   struct Complex result;
   result.real = a.real - b.real;
   result.imaginary = a.imaginary - b.imaginary;
   return result;
}

// Function to multiply two complex numbers
struct Complex multiply(struct Complex a, struct Complex b) {
   struct Complex result;
   result.real = a.real * b.real - a.imaginary * b.imaginary;
   result.imaginary = a.real * b.imaginary + a.imaginary * b.real;
   return result;
}

int main() {
   struct Complex num1, num2, result;

   // Input two complex numbers
   printf("Enter first complex number (real and imaginary): ");
   scanf("%f %f", &num1.real, &num1.imaginary);

   printf("Enter second complex number (real and imaginary): ");
   scanf("%f %f", &num2.real, &num2.imaginary);

   // Perform operations
   result = add(num1, num2);
   printf("Sum: %.2f + %.2fi\n", result.real, result.imaginary);

   result = subtract(num1, num2);
   printf("Difference: %.2f + %.2fi\n", result.real, result.imaginary);

   result = multiply(num1, num2);
   printf("Product: %.2f + %.2fi\n", result.real, result.imaginary);

   return 0;
}
```

```c
#include <stdio.h>

// Define structure for a bank account
struct BankAccount {
    int accountNumber;
    char accountHolderName[100];
    float balance;
};

// Function to deposit money
void deposit(struct BankAccount *account, float amount) {
    if (amount > 0) {
        account->balance += amount;
        printf("Deposited %.2f. New balance: %.2f\n", amount, account->balance);
    } else {
        printf("Invalid deposit amount.\n");
    }
}

// Function to withdraw money
void withdraw(struct BankAccount *account, float amount) {
    if (amount > 0 && amount <= account->balance) {
        account->balance -= amount;
        printf("Withdrawn %.2f. New balance: %.2f\n", amount, account->balance);
    } else {
        printf("Invalid withdrawal amount or insufficient balance.\n");
    }
}

// Main function to demonstrate deposit and withdrawal
```

```
int main() {
    struct BankAccount account = {123456, "John Doe", 1000.0};

    printf("Account Holder: %s\nAccount Number: %d\nInitial Balance: %.2f\n\n",
          account.accountHolderName, account.accountNumber, account.balance);

    deposit(&account, 500.0);
    withdraw(&account, 200.0);
    withdraw(&account, 1500.0);

    return 0;
}
```

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

// Define structure for a car
struct Car {
    char make[50];
    char model[50];
    int year;
    float price;
};

// Function to print cars within a specified price range
void printCarsWithinPriceRange(struct Car cars[], int size, float minPrice, float maxPrice) {
    printf("\nCars within the price range %.2f to %.2f:\n", minPrice, maxPrice);
    for (int i = 0; i < size; i++) {
        if (cars[i].price >= minPrice && cars[i].price <= maxPrice) {
            printf("Make: %s, Model: %s, Year: %d, Price: %.2f\n",
                  cars[i].make, cars[i].model, cars[i].year, cars[i].price);
        }
    }
}

int main() {
    struct Car inventory[5];

    // Input car details
    for (int i = 0; i < 5; i++) {
        printf("\nEnter details for car %d:\n", i + 1);
        printf("Make: ");
        scanf("%s", inventory[i].make);
        printf("Model: ");
        scanf("%s", inventory[i].model);
        printf("Year: ");
        scanf("%d", &inventory[i].year);
        printf("Price: ");
        scanf("%f", &inventory[i].price);
    }

    // Specify price range and print matching cars
    float minPrice, maxPrice;
    printf("\nEnter the minimum price: ");
    scanf("%f", &minPrice);
    printf("Enter the maximum price: ");
    scanf("%f", &maxPrice);

    printCarsWithinPriceRange(inventory, 5, minPrice, maxPrice);

    return 0;}
```

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

// Define a structure to represent a library book
struct Book {
   char title[100];
   char author[100];
   int publication_year;
   char status[10];  // "available" or "issued"
};

// Function to issue a book
void issueBook(struct Book* book) {
   if (strcmp(book->status, "available") == 0) {
      strcpy(book->status, "issued");
      printf("Book '%s' has been issued.\n", book->title);
   } else {
      printf("Sorry, the book '%s' is already issued.\n", book->title);
   }
}

// Function to return a book
void returnBook(struct Book* book) {
   if (strcmp(book->status, "issued") == 0) {
      strcpy(book->status, "available");
      printf("Book '%s' has been returned.\n", book->title);
   } else {
      printf("The book '%s' was not issued.\n", book->title);
   }
```

```
}

// Function to display book details
void displayBook(struct Book book) {
    printf("Title: %s\n", book.title);
    printf("Author: %s\n", book.author);
    printf("Publication Year: %d\n", book.publication_year);
    printf("Status: %s\n", book.status);
}

int main() {
    // Create a book instance
    struct Book book1 = {"The Great Gatsby", "F. Scott Fitzgerald", 1925, "available"};

    // Display initial details
    printf("Initial Book Details:\n");
    displayBook(book1);

    // Issue the book
    issueBook(&book1);

    // Try to issue the same book again
    issueBook(&book1);

    // Return the book
    returnBook(&book1);

    // Display final details
    printf("\nFinal Book Details:\n");
    displayBook(book1);
    return 0;
}
```

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

// Define a structure to represent a student
struct Student {
    char name[100];
    int roll_number;
    int grades[5];  // Array to store grades (assuming 5 subjects for simplicity)
};

// Function to calculate the highest grade
int calculateHighestGrade(struct Student student) {
    int highest = student.grades[0];
    for (int i = 1; i < 5; i++) {
        if (student.grades[i] > highest) {
            highest = student.grades[i];
        }
    }
    return highest;
}

// Function to calculate the lowest grade
int calculateLowestGrade(struct Student student) {
    int lowest = student.grades[0];
    for (int i = 1; i < 5; i++) {
        if (student.grades[i] < lowest) {
            lowest = student.grades[i];
        }
    }
    return lowest;
}

// Function to calculate the average grade
float calculateAverageGrade(struct Student student) {
    int total = 0;
    for (int i = 0; i < 5; i++) {
        total += student.grades[i];
    }
    return total / 5.0;
}

// Function to display student information and grade statistics
void displayStudentInfo(struct Student student) {
    printf("Student Name: %s\n", student.name);
    printf("Roll Number: %d\n", student.roll_number);

    // Display grades
    printf("Grades: ");
    for (int i = 0; i < 5; i++) {
        printf("%d ", student.grades[i]);
    }
    printf("\n");

    // Display highest, lowest, and average grade
    printf("Highest Grade: %d\n", calculateHighestGrade(student));
    printf("Lowest Grade: %d\n", calculateLowestGrade(student));
    printf("Average Grade: %.2f\n", calculateAverageGrade(student));
```

```
}

int main() {
    // Create a student instance
    struct Student student1 = {"John Doe", 101, {85, 92, 78, 88, 90}};

    // Display student information and grade statistics
    displayStudentInfo(student1);

    return 0;
}
```

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

// Define a structure to represent a product
struct Product {
    int product_id;       // Unique ID for the product
    char name[100];       // Product name
    int quantity;         // Quantity available in stock
    float price;          // Price of the product
};

// Function to update the quantity after a sale
void updateQuantity(struct Product* product, int sold_quantity) {
    if (sold_quantity <= product->quantity) {
        product->quantity -= sold_quantity;
        printf("Sale successful! Sold %d units of '%s'.\n", sold_quantity, product->name);
    } else {
        printf("Not enough stock available for '%s'. Only %d units left.\n", product->name, product-
>quantity);
    }
}

// Function to calculate the total sales value for a product sold
float calculateTotalSalesValue(struct Product product, int sold_quantity) {
    if (sold_quantity <= product.quantity) {
        return sold_quantity * product.price;
    } else {
        return 0;  // If not enough stock is available, return 0
    }
}

// Function to display product information
void displayProduct(struct Product product) {
    printf("Product ID: %d\n", product.product_id);
    printf("Product Name: %s\n", product.name);
    printf("Quantity Available: %d\n", product.quantity);
    printf("Price per Unit: $%.2f\n", product.price);
}

int main() {
    // Create a product instance
    struct Product product1 = {101, "Laptop", 50, 799.99};

    // Display product details
    displayProduct(product1);

    // Let's sell 5 units of the product
    int sold_quantity = 5;
    float total_sales_value = calculateTotalSalesValue(product1, sold_quantity);


    if (total_sales_value > 0) {
        printf("Total Sales Value for selling %d units: $%.2f\n", sold_quantity, total_sales_value);
    }


    // Update product quantity after sale
    updateQuantity(&product1, sold_quantity);


    // Display updated product details
    printf("\nUpdated Product Details:\n");
    displayProduct(product1);


    return 0;
}
```

Additional Problem Statements of the structure:

1. **Point Distance Calculation**:
   - Define a structure for a point in 2D space (x, y).
   - Write a function to calculate the distance between two points.
2. **Rectangle Properties**:
   - Create a structure for a rectangle with length and width.
   - Write functions to calculate the area and perimeter of the rectangle.
3. **Movie Details**:
   - Define a structure to store details of a movie, including title, director, release year, and rating.
   - Write a program to sort movies by their rating.
4. **Weather Report**:
   - Create a structure to store daily weather data, including date, temperature, and humidity.

    - Write a program to find the day with the highest temperature.

5. **Fraction Arithmetic**:
    - Define a structure for a fraction with numerator and denominator.
    - Write functions to add, subtract, multiply, and divide two fractions.
6. **Laptop Inventory**:
    - Create a structure to represent a laptop with fields for brand, model, processor, RAM, and price.
    - Write a program to list laptops within a specific price range.
7. **Student Attendance**:
    - Define a structure to store attendance data, including student ID, total classes, and classes attended.
    - Write a program to calculate and display the attendance percentage for each student.
8. **Flight Information**:
    - Create a structure for a flight with fields for flight number, departure, destination, and duration.
    - Write a program to display flights that are less than a specified duration.
9. **Polynomial Representation**:
    - Define a structure to represent a term of a polynomial (coefficient and exponent).
    - Write functions to add and multiply two polynomials.
10. **Medical Records**:
    - Create a structure for a patient's medical record with fields for name, age, diagnosis, and treatment.
    - Write a program to search for patients by diagnosis.
11. **Game Scores**:
    - Define a structure to store player information, including name, game played, and score.
    - Write a program to display the top scorer for each game.
12. **City Information**:
    - Create a structure to store information about a city, including name, population, and area.
    - Write a program to calculate and display the population density of each city.
13. **Vehicle Registration**:
    - Define a structure for vehicle registration details, including registration number, owner, make, and year.
    - Write a program to list all vehicles registered in a given year.
14. **Restaurant Menu**:
    - Create a structure to represent a menu item with fields for name, category, and price.
    - Write a program to display menu items in a specific category.
15. **Sports Team**:
    - Define a structure for a sports team with fields for team name, sport, number of players, and coach.
    - Write a program to display all teams playing a specific sport.
16. **Student Marks Analysis**:
    - Create a structure to store student marks in different subjects.
    - Write a program to calculate the total and percentage of marks for each student.
17. **E-commerce Product**:

- Define a structure for an e-commerce product with fields for product ID, name, category, price, and stock.
- Write a program to update the stock and calculate the total value of products in stock.

18. **Music Album**:
    - Create a structure to store details of a music album, including album name, artist, genre, and release year.
    - Write a program to display albums of a specific genre.
19. **Cinema Ticket Booking**:
    - Define a structure for a cinema ticket with fields for movie name, seat number, and price.
    - Write a program to book tickets and display the total revenue generated.
20. **University Courses**:
    - Create a structure to store course details, including course code, name, instructor, and credits.
    - Write a program to list all courses taught by a specific instructor.

```
#include <stdio.h>

#include <math.h>


typedef struct {

   float x;

   float y;

} Point;


float calculate_distance(Point p1, Point p2) {

   return sqrt(pow(p1.x - p2.x, 2) + pow(p1.y - p2.y, 2));

}


int main() {

   Point p1 = {1, 2};

   Point p2 = {4, 6};

   printf("Distance: %.2f\n", calculate_distance(p1, p2));

   return 0;

}
```

```
#include <stdio.h>

typedef struct {
    float length;
    float width;
} Rectangle;

float calculate_area(Rectangle r) {
    return r.length * r.width;
}

float calculate_perimeter(Rectangle r) {
    return 2 * (r.length + r.width);
}

int main() {
    Rectangle rect = {5, 3};
    printf("Area: %.2f\n", calculate_area(rect));
    printf("Perimeter: %.2f\n", calculate_perimeter(rect));
    return 0;
}
```

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

typedef struct {
    char title[50];
    char director[50];
    int release_year;
    float rating;
} Movie;
```

```
void sort_movies_by_rating(Movie movies[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (movies[j].rating < movies[j + 1].rating) {
                Movie temp = movies[j];
                movies[j] = movies[j + 1];
                movies[j + 1] = temp;
            }
        }
    }
}

int main() {
    Movie movies[2] = {
        {"Inception", "Christopher Nolan", 2010, 8.8},
        {"Interstellar", "Christopher Nolan", 2014, 8.6}
    };

    sort_movies_by_rating(movies, 2);

    printf("Movies sorted by rating:\n");
    for (int i = 0; i < 2; i++) {
        printf("%s - %.1f\n", movies[i].title, movies[i].rating);
    }
    return 0;
}
```

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

```c
typedef struct {
    char date[11];
    float temperature;
    int humidity;
} Weather;

Weather highest_temperature(Weather data[], int n) {
    Weather hottest = data[0];
    for (int i = 1; i < n; i++) {
        if (data[i].temperature > hottest.temperature) {
            hottest = data[i];
        }
    }
    return hottest;
}

int main() {
    Weather weather_data[2] = {
        {"2023-01-01", 30.5, 60},
        {"2023-01-02", 35.2, 55}
    };

    Weather hottest = highest_temperature(weather_data, 2);
    printf("Hottest Day: %s with %.1f°C\n", hottest.date, hottest.temperature);
    return 0;
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
```

```
typedef struct {
    int numerator;
    int denominator;
} Fraction;

int gcd(int a, int b) {
    return b == 0 ? a : gcd(b, a % b);
}

Fraction simplify(Fraction f) {
    int common_factor = gcd(f.numerator, f.denominator);
    f.numerator /= common_factor;
    f.denominator /= common_factor;
    return f;
}

Fraction add(Fraction f1, Fraction f2) {
    Fraction result;
    result.numerator = f1.numerator * f2.denominator + f2.numerator * f1.denominator;
    result.denominator = f1.denominator * f2.denominator;
    return simplify(result);
}

Fraction subtract(Fraction f1, Fraction f2) {
    Fraction result;
    result.numerator = f1.numerator * f2.denominator - f2.numerator * f1.denominator;
    result.denominator = f1.denominator * f2.denominator;
    return simplify(result);
}
```

```
Fraction multiply(Fraction f1, Fraction f2) {
    Fraction result;
    result.numerator = f1.numerator * f2.numerator;
    result.denominator = f1.denominator * f2.denominator;
    return simplify(result);
}

Fraction divide(Fraction f1, Fraction f2) {
    if (f2.numerator == 0) {
        fprintf(stderr, "Error: Division by zero.\n");
        exit(EXIT_FAILURE);
    }
    Fraction result;
    result.numerator = f1.numerator * f2.denominator;
    result.denominator = f1.denominator * f2.numerator;
    return simplify(result);
}

void printFraction(Fraction f) {
    printf("%d/%d\n", f.numerator, f.denominator);
}

int main() {
    Fraction f1 = {1, 2};
    Fraction f2 = {3, 4};

    printf("Addition: ");
    printFraction(add(f1, f2));
    printf("Subtraction: ");
    printFraction(subtract(f1, f2));

    printf("Multiplication: ");
    printFraction(multiply(f1, f2));

    printf("Division: ");
    printFraction(divide(f1, f2));

    return 0;
}
```

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

typedef struct {
    char brand[50];
    char model[50];
    char processor[50];
    int RAM;
    float price;
} Laptop;

void printLaptopInfo(Laptop laptop) {
    printf("Brand: %s\n", laptop.brand);
    printf("Model: %s\n", laptop.model);
    printf("Processor: %s\n", laptop.processor);
    printf("RAM: %d GB\n", laptop.RAM);
    printf("Price: $%.2f\n", laptop.price);
}
```

```
int main() {
    Laptop laptop1;

    strcpy(laptop1.brand, "Dell");
    strcpy(laptop1.model, "XPS 15");
    strcpy(laptop1.processor, "Intel Core i7");
    laptop1.RAM = 16;
    laptop1.price = 1499.99;

    printf("Laptop Inventory:\n");
    printLaptopInfo(laptop1);

    return 0;
}
```

```
#include <stdio.h>

typedef struct {
    int studentID;
    int totalClasses;
    int classesAttended;
} Attendance;

float calculateAttendancePercentage(Attendance student) {
    if (student.totalClasses == 0) {
        return 0.0;
    }
    return ((float)student.classesAttended / student.totalClasses) * 100;
}

void displayAttendance(Attendance student) {
    float percentage = calculateAttendancePercentage(student);
    printf("Student ID: %d\n", student.studentID);
    printf("Total Classes: %d\n", student.totalClasses);
    printf("Classes Attended: %d\n", student.classesAttended);
    printf("Attendance Percentage: %.2f%%\n", percentage);
}

int main() {
    Attendance students[3] = {
        {101, 50, 45},
        {102, 50, 40},
        {103, 50, 48}
    };

    printf("Student Attendance Records:\n");
    for (int i = 0; i < 3; i++) {
        displayAttendance(students[i]);
        printf("\n");
    }

    return 0;
}
```

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

typedef struct {
    char flightNumber[10];
    char departure[50];
    char destination[50];
    int duration; // in minutes
} Flight;

void displayShortFlights(Flight flights[], int size, int maxDuration) {
    printf("Flights with duration less than %d minutes:\n", maxDuration);
    for (int i = 0; i < size; i++) {
        if (flights[i].duration < maxDuration) {
            printf("Flight Number: %s, Departure: %s, Destination: %s, Duration: %d minutes\n",
                   flights[i].flightNumber, flights[i].departure, flights[i].destination, flights[i].duration);
        }
    }
}

int main() {
    Flight flights[3] = {
        {"AA123", "New York", "Los Angeles", 300},
        {"BA456", "London", "Paris", 75},
        {"CA789", "Tokyo", "Osaka", 90}
    };

    int maxDuration = 100;
    displayShortFlights(flights, 3, maxDuration);

    return 0;
}
```

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

typedef struct {
    int coefficient;
    int exponent;
```

```
} Term;

void addPolynomials(Term poly1[], int size1, Term poly2[], int size2) {
    int i = 0, j = 0;
    printf("Sum of Polynomials:\n");
    while (i < size1 && j < size2) {
        if (poly1[i].exponent > poly2[j].exponent) {
            printf("%dx^%d + ", poly1[i].coefficient, poly1[i].exponent);
            i++;
        } else if (poly1[i].exponent < poly2[j].exponent) {
            printf("%dx^%d + ", poly2[j].coefficient, poly2[j].exponent);
            j++;
        } else {
            printf("%dx^%d + ", poly1[i].coefficient + poly2[j].coefficient, poly1[i].exponent);
            i++;
            j++;
        }
    }
    while (i < size1) {
        printf("%dx^%d + ", poly1[i].coefficient, poly1[i].exponent);
        i++;
    }
    while (j < size2) {
        printf("%dx^%d + ", poly2[j].coefficient, poly2[j].exponent);
        j++;
    }
    printf("\n");
}

void multiplyPolynomials(Term poly1[], int size1, Term poly2[], int size2) {
    printf("Product of Polynomials:\n");
    for (int i = 0; i < size1; i++) {
        for (int j = 0; j < size2; j++) {
            printf("%dx^%d + ", poly1[i].coefficient * poly2[j].coefficient,
                   poly1[i].exponent + poly2[j].exponent);
        }
    }
    printf("\n");
}

int main() {
    Term poly1[] = {{3, 2}, {5, 1}, {6, 0}};
    Term poly2[] = {{4, 2}, {2, 1}};

    int size1 = sizeof(poly1) / sizeof(poly1[0]);
    int size2 = sizeof(poly2) / sizeof(poly2[0]);

    addPolynomials(poly1, size1, poly2, size2);
    multiplyPolynomials(poly1, size1, poly2, size2);

    return 0;
}
```

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

typedef struct {
    char name[50];
    int age;
    char diagnosis[50];
    char treatment[100];
} MedicalRecord;


void searchByDiagnosis(MedicalRecord records[], int size, const char* diagnosis) {
    printf("Patients with diagnosis '%s':\n", diagnosis);
    for (int i = 0; i < size; i++) {
        if (strcmp(records[i].diagnosis, diagnosis) == 0) {
            printf("Name: %s, Age: %d, Treatment: %s\n", records[i].name, records[i].age,
records[i].treatment);
        }
    }
}


int main() {
    MedicalRecord records[3] = {
        {"John Doe", 45, "Flu", "Rest and hydration"},
        {"Jane Smith", 30, "Cold", "Over-the-counter medication"},
        {"Alice Brown", 50, "Flu", "Antiviral medication"}
    };


    searchByDiagnosis(records, 3, "Flu");


    return 0;
}
```

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


#define MAX_PLAYERS 100
#define MAX_GAMES 5
#define NAME_LEN 50
```

```
// Define a structure to store player information
struct Player {
    char name[NAME_LEN];
    char game[NAME_LEN];
    int score;
};

void findTopScorer(struct Player players[], int playerCount, char game[]) {
    int topScore = -1;
    char topPlayer[NAME_LEN];

    // Find the top scorer for the given game
    for (int i = 0; i < playerCount; i++) {
        if (strcmp(players[i].game, game) == 0) { // Match the game
            if (players[i].score > topScore) {
                topScore = players[i].score;
                strcpy(topPlayer, players[i].name);
            }
        }
    }

    if (topScore != -1) {
        printf("Top scorer for %s: %s with score %d\n", game, topPlayer, topScore);
    } else {
        printf("No players found for the game: %s\n", game);
    }
}

int main() {
    int playerCount;
    // Input number of players
    printf("Enter the number of players: ");
    scanf("%d", &playerCount);

    // Array to store player data
    struct Player players[MAX_PLAYERS];

    // Input player information
    for (int i = 0; i < playerCount; i++) {
        printf("Enter name of player %d: ", i + 1);
        scanf("%s", players[i].name);

        printf("Enter the game played by player %d: ", i + 1);
        scanf("%s", players[i].game);

        printf("Enter score for player %d: ", i + 1);
        scanf("%d", &players[i].score);
    }

    // Find and display the top scorer for each game
    char games[MAX_GAMES][NAME_LEN] = {"Football", "Basketball", "Tennis", "Baseball", "Cricket"};

    for (int i = 0; i < MAX_GAMES; i++) {
        printf("\nChecking for top scorer in %s:\n", games[i]);
        findTopScorer(players, playerCount, games[i]);
    }

    return 0;
}
```

```c
#include <stdio.h>

#define MAX_CITIES 100
#define NAME_LEN 50

// Define a structure to store city information
struct City {
    char name[NAME_LEN];
    int population;
    float area;
};

// Function to calculate and display population density
void displayPopulationDensity(struct City cities[], int cityCount) {
    for (int i = 0; i < cityCount; i++) {
        if (cities[i].area > 0) {
            float density = cities[i].population / cities[i].area;
            printf("City: %s\n", cities[i].name);
            printf("Population: %d\n", cities[i].population);
            printf("Area: %.2f square kilometers\n", cities[i].area);
            printf("Population Density: %.2f people per square kilometer\n\n", density);
        } else {
            printf("Invalid area for city %s\n\n", cities[i].name);
        }
    }
}

int main() {
    int cityCount;
    // Input number of cities
    printf("Enter the number of cities: ");
    scanf("%d", &cityCount);

    // Array to store city information
    struct City cities[MAX_CITIES];

    // Input city details
    for (int i = 0; i < cityCount; i++) {
        printf("Enter name of city %d: ", i + 1);
        scanf("%s", cities[i].name);

        printf("Enter population of city %d: ", i + 1);
        scanf("%d", &cities[i].population);

        printf("Enter area (in square kilometers) of city %d: ", i + 1);
        scanf("%f", &cities[i].area);
    }

    // Display population density for each city
    displayPopulationDensity(cities, cityCount);

    return 0;
}
```

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

#define MAX_VEHICLES 100
#define NAME_LEN 50
#define REG_LEN 20
```

```
#define MAKE_LEN 30

// Define a structure for vehicle registration details
struct Vehicle {
    char registrationNumber[REG_LEN];
    char owner[NAME_LEN];
    char make[MAKE_LEN];
    int year;
};

// Function to list all vehicles registered in a given year
void listVehiclesByYear(struct Vehicle vehicles[], int vehicleCount, int year) {
    int found = 0;

    // Loop through all vehicles and check their registration year
    for (int i = 0; i < vehicleCount; i++) {
        if (vehicles[i].year == year) {
            printf("Registration Number: %s\n", vehicles[i].registrationNumber);
            printf("Owner: %s\n", vehicles[i].owner);
            printf("Make: %s\n", vehicles[i].make);
            printf("Year: %d\n\n", vehicles[i].year);
            found = 1;
        }
    }

    // If no vehicles found for the given year
    if (!found) {
        printf("No vehicles registered in the year %d.\n", year);
    }
}
```

```
int main() {
  int vehicleCount;

  // Input number of vehicles
  printf("Enter the number of vehicles: ");
  scanf("%d", &vehicleCount);

  // Array to store vehicle details
  struct Vehicle vehicles[MAX_VEHICLES];

  // Input vehicle details
  for (int i = 0; i < vehicleCount; i++) {
    printf("Enter registration number of vehicle %d: ", i + 1);
    scanf("%s", vehicles[i].registrationNumber);

    printf("Enter owner name of vehicle %d: ", i + 1);
    scanf("%s", vehicles[i].owner);

    printf("Enter make of vehicle %d: ", i + 1);
    scanf("%s", vehicles[i].make);

    printf("Enter year of registration for vehicle %d: ", i + 1);
    scanf("%d", &vehicles[i].year);
  }

  int searchYear;

  // Ask for the year to list vehicles registered in that year
  printf("Enter the year to search for registered vehicles: ");
```

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

    // List vehicles registered in the given year
    listVehiclesByYear(vehicles, vehicleCount, searchYear);

    return 0;
}
```

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

#define MAX_MENU_ITEMS 100
#define NAME_LEN 50
#define CATEGORY_LEN 30

// Define a structure for a menu item
struct MenuItem {
    char name[NAME_LEN];
    char category[CATEGORY_LEN];
    float price;
};

// Function to display the menu
void displayMenu(struct MenuItem menu[], int itemCount) {
    printf("\nRestaurant Menu:\n");
    printf("-------------------\n");
    for (int i = 0; i < itemCount; i++) {
        printf("Name: %s\n", menu[i].name);
        printf("Category: %s\n", menu[i].category);
        printf("Price: $%.2f\n", menu[i].price);
        printf("-------------------\n");
    }
}

int main() {
    int itemCount;

    // Input number of menu items
    printf("Enter the number of menu items: ");
    scanf("%d", &itemCount);

    // Array to store menu items
    struct MenuItem menu[MAX_MENU_ITEMS];

    // Input menu item details
    for (int i = 0; i < itemCount; i++) {
        printf("Enter name of menu item %d: ", i + 1);
        scanf(" %[^\n]s", menu[i].name); // Reads the full line including spaces

        printf("Enter category of menu item %d: ", i + 1);
        scanf(" %[^\n]s", menu[i].category);

        printf("Enter price of menu item %d: ", i + 1);
        scanf("%f", &menu[i].price);
    }

    // Display the menu
    displayMenu(menu, itemCount);

    return 0;
}
```

```
#include <stdio.h>

#include <string.h>


#define MAX_TEAMS 100

#define NAME_LEN 50

#define SPORT_LEN 30

#define COACH_LEN 50


// Define a structure for a sports team

struct Team {

    char teamName[NAME_LEN];

    char sport[SPORT_LEN];

    int numPlayers;

    char coach[COACH_LEN];

};


// Function to display all teams playing a specific sport

void displayTeamsBySport(struct Team teams[], int teamCount, const char *sport) {

    int found = 0;


    // Loop through all teams and check if their sport matches the input sport

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

        if (strcmp(teams[i].sport, sport) == 0) {

            printf("Team Name: %s\n", teams[i].teamName);

            printf("Sport: %s\n", teams[i].sport);

            printf("Number of Players: %d\n", teams[i].numPlayers);

            printf("Coach: %s\n\n", teams[i].coach);

            found = 1;

        }

    }
    // If no teams are found for the specified sport
    if (!found) {
        printf("No teams found for the sport %s.\n", sport);
    }
}

int main() {
    int teamCount;

    // Input number of teams
    printf("Enter the number of sports teams: ");
    scanf("%d", &teamCount);

    // Array to store team details
    struct Team teams[MAX_TEAMS];

    // Input details for each team
    for (int i = 0; i < teamCount; i++) {
        printf("Enter team name for team %d: ", i + 1);
        scanf(" %[^\n]s", teams[i].teamName);  // Read full line including spaces

        printf("Enter sport for team %d: ", i + 1);
        scanf(" %[^\n]s", teams[i].sport);  // Read full line for sport

        printf("Enter number of players for team %d: ", i + 1);
        scanf("%d", &teams[i].numPlayers);

        printf("Enter coach name for team %d: ", i + 1);
        scanf(" %[^\n]s", teams[i].coach);  // Read full line for coach name
    }

    // Ask for a sport to display teams that play it
    char sportToSearch[SPORT_LEN];
    printf("Enter the sport to search for: ");
    scanf(" %[^\n]s", sportToSearch);  // Read sport with spaces

    // Display all teams that play the specified sport
    displayTeamsBySport(teams, teamCount, sportToSearch);

    return 0;
}
```

```
#include <stdio.h>

#define MAX_SUBJECTS 5
#define NAME_LEN 50

// Define a structure to store student marks
struct Student {
    char name[NAME_LEN];
    int marks[MAX_SUBJECTS];
    int total;
    float percentage;
};

// Function to calculate the total and percentage
void calculateTotalAndPercentage(struct Student *student, int numSubjects) {
    student->total = 0;

    // Calculate total marks
    for (int i = 0; i < numSubjects; i++) {
        student->total += student->marks[i];
    }

    // Calculate percentage
    student->percentage = ((float)student->total / (numSubjects * 100)) * 100;
}

// Function to display student details
void displayStudentDetails(struct Student student, int numSubjects) {
    printf("\nStudent Name: %s\n", student.name);
    printf("Marks: ");
    for (int i = 0; i < numSubjects; i++) {
        printf("%d ", student.marks[i]);
    }
    printf("\nTotal Marks: %d\n", student.total);
    printf("Percentage: %.2f%%\n", student.percentage);
}

int main() {
    int numStudents, numSubjects = MAX_SUBJECTS;

    // Input number of students
    printf("Enter the number of students: ");
    scanf("%d", &numStudents);

    // Array to store student details
    struct Student students[numStudents];

    // Input details for each student
    for (int i = 0; i < numStudents; i++) {
        printf("\nEnter details for student %d:\n", i + 1);

        // Input student name
        printf("Enter student name: ");
        scanf(" %[^\n]s", students[i].name);  // Read full name

        // Input marks for each subject
        printf("Enter marks for %d subjects (out of 100 each):\n", numSubjects);
        for (int j = 0; j < numSubjects; j++) {
            printf("Subject %d: ", j + 1);
            scanf("%d", &students[i].marks[j]);
        }

        // Calculate total and percentage
        calculateTotalAndPercentage(&students[i], numSubjects);
    }

    // Display the details of all students
    for (int i = 0; i < numStudents; i++) {
        displayStudentDetails(students[i], numSubjects);
    }

    return 0;
}
```

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

#define MAX_PRODUCTS 100
#define NAME_LEN 50
```

```
#define CATEGORY_LEN 30

// Define a structure for an e-commerce product
struct Product {
   int productID;
   char name[NAME_LEN];
   char category[CATEGORY_LEN];
   float price;
   int stock;
};

// Function to update the stock of a product
void updateStock(struct Product *product, int newStock) {
   product->stock = newStock;
}

// Function to calculate the total value of products in stock
float calculateTotalValue(struct Product product) {
   return product.price * product.stock;
}

// Function to display product details
void displayProductDetails(struct Product product) {
   printf("\nProduct ID: %d\n", product.productID);
   printf("Name: %s\n", product.name);
   printf("Category: %s\n", product.category);
   printf("Price: $%.2f\n", product.price);
   printf("Stock: %d\n", product.stock);
   printf("Total Value of Stock: $%.2f\n", calculateTotalValue(product));
}
```

```
int main() {
   int numProducts;

   // Input the number of products
   printf("Enter the number of products: ");
   scanf("%d", &numProducts);

   // Array to store product details
   struct Product products[MAX_PRODUCTS];

   // Input details for each product
   for (int i = 0; i < numProducts; i++) {
      printf("\nEnter details for product %d:\n", i + 1);

      printf("Enter product ID: ");
      scanf("%d", &products[i].productID);

      printf("Enter product name: ");
      scanf(" %[^\n]s", products[i].name);  // Read the full name with spaces

      printf("Enter product category: ");
      scanf(" %[^\n]s", products[i].category);  // Read the full category name with spaces

      printf("Enter product price: ");
      scanf("%f", &products[i].price);

      printf("Enter product stock: ");
      scanf("%d", &products[i].stock);
   }
```

```
// Ask user to update the stock for a specific product
int productIDToUpdate, newStock;
printf("\nEnter product ID to update stock: ");
scanf("%d", &productIDToUpdate);
printf("Enter new stock quantity: ");
scanf("%d", &newStock);

// Update stock for the specified product
int found = 0;
for (int i = 0; i < numProducts; i++) {
    if (products[i].productID == productIDToUpdate) {
        updateStock(&products[i], newStock);
        printf("\nStock updated for product %d!\n", productIDToUpdate);
        found = 1;
        break;
    }
}

// If the product ID was not found
if (!found) {
    printf("\nProduct ID %d not found.\n", productIDToUpdate);
}

// Display the details of all products
for (int i = 0; i < numProducts; i++) {
    displayProductDetails(products[i]);
}

return 0;
```

```
}
```

```
#include <stdio.h>

#define MAX_ALBUMS 100
#define NAME_LEN 100
#define ARTIST_LEN 50
#define GENRE_LEN 30

// Define a structure for a music album
struct MusicAlbum {
    char albumName[NAME_LEN];
    char artist[ARTIST_LEN];
    char genre[GENRE_LEN];
    int releaseYear;
};

// Function to display details of a music album
void displayAlbumDetails(struct MusicAlbum album) {
    printf("\nAlbum Name: %s\n", album.albumName);
    printf("Artist: %s\n", album.artist);
    printf("Genre: %s\n", album.genre);
    printf("Release Year: %d\n", album.releaseYear);
}

int main() {
    int numAlbums;

    // Input number of albums
    printf("Enter the number of albums: ");
    scanf("%d", &numAlbums);
    // Array to store album details
    struct MusicAlbum albums[MAX_ALBUMS];

    // Input details for each album
    for (int i = 0; i < numAlbums; i++) {
        printf("\nEnter details for album %d:\n", i + 1);

        printf("Enter album name: ");
        scanf(" %[^\n]s", albums[i].albumName);  // Read full name with spaces

        printf("Enter artist name: ");
        scanf(" %[^\n]s", albums[i].artist);  // Read full artist name with spaces

        printf("Enter genre: ");
        scanf(" %[^\n]s", albums[i].genre);  // Read full genre name with spaces

        printf("Enter release year: ");
        scanf("%d", &albums[i].releaseYear);
    }

    // Display details of all albums
    for (int i = 0; i < numAlbums; i++) {
        displayAlbumDetails(albums[i]);
    }

    return 0;
}
```

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

```
#define MAX_TICKETS 100

#define MOVIE_NAME_LEN 100


// Define a structure for a cinema ticket

struct CinemaTicket {

    char movieName[MOVIE_NAME_LEN];

    int seatNumber;

    float price;

};


// Function to book tickets and calculate total revenue

void bookTicket(struct CinemaTicket *ticket) {

    printf("Enter movie name: ");

    scanf(" %[^\n]s", ticket->movieName);  // Read full movie name with spaces


    printf("Enter seat number: ");

    scanf("%d", &ticket->seatNumber);


    printf("Enter ticket price: ");

    scanf("%f", &ticket->price);

}


// Function to display ticket details

void displayTicketDetails(struct CinemaTicket ticket) {

    printf("\nMovie Name: %s\n", ticket.movieName);

    printf("Seat Number: %d\n", ticket.seatNumber);

    printf("Price: $%.2f\n", ticket.price);

}


int main() {
```

```
int numTickets;
float totalRevenue = 0.0;

// Input the number of tickets to be booked
printf("Enter the number of tickets to be booked: ");
scanf("%d", &numTickets);

// Array to store ticket details
struct CinemaTicket tickets[MAX_TICKETS];

// Book tickets and calculate total revenue
for (int i = 0; i < numTickets; i++) {
   printf("\nEnter details for ticket %d:\n", i + 1);

   // Book a ticket
   bookTicket(&tickets[i]);

   // Add the price of the ticket to the total revenue
   totalRevenue += tickets[i].price;
}

// Display the details of all booked tickets
printf("\n--- Ticket Details ---\n");
for (int i = 0; i < numTickets; i++) {
   displayTicketDetails(tickets[i]);
}

// Display total revenue generated
printf("\nTotal Revenue Generated: $%.2f\n", totalRevenue);
```

```
    return 0;
}
```

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

#define MAX_COURSES 100
#define COURSE_NAME_LEN 100
#define INSTRUCTOR_NAME_LEN 100

// Define a structure for a university course
struct Course {
    char courseCode[10];
    char courseName[COURSE_NAME_LEN];
    char instructor[INSTRUCTOR_NAME_LEN];
    int credits;
};

// Function to input course details
void inputCourseDetails(struct Course *course) {
    printf("Enter course code: ");
    scanf("%s", course->courseCode);

    printf("Enter course name: ");
    scanf(" %[^\n]s", course->courseName);  // Read full course name with spaces

    printf("Enter instructor name: ");
    scanf(" %[^\n]s", course->instructor);  // Read full instructor name with spaces

    printf("Enter number of credits: ");
    scanf("%d", &course->credits);
```

```c
}

// Function to list courses taught by a specific instructor
void listCoursesByInstructor(struct Course courses[], int numCourses, char instructor[]) {
    printf("\nCourses taught by %s:\n", instructor);
    int found = 0;
    for (int i = 0; i < numCourses; i++) {
        if (strcmp(courses[i].instructor, instructor) == 0) {
            printf("Course Code: %s\n", courses[i].courseCode);
            printf("Course Name: %s\n", courses[i].courseName);
            printf("Credits: %d\n", courses[i].credits);
            printf("----------------------------\n");
            found = 1;
        }
    }
    if (!found) {
        printf("No courses found for instructor %s.\n", instructor);
    }
}

int main() {
    int numCourses;
    char instructorToSearch[INSTRUCTOR_NAME_LEN];

    // Input the number of courses
    printf("Enter the number of courses: ");
    scanf("%d", &numCourses);

    // Array to store course details
    struct Course courses[MAX_COURSES];
    // Input details for each course
    for (int i = 0; i < numCourses; i++) {
        printf("\nEnter details for course %d:\n", i + 1);
        inputCourseDetails(&courses[i]);
    }

    // Input instructor name to search for
    printf("\nEnter instructor name to list their courses: ");
    scanf(" %[^\n]s", instructorToSearch);  // Read full instructor name with spaces

    // List all courses taught by the specified instructor
    listCoursesByInstructor(courses, numCourses, instructorToSearch);

    return 0;
}
```