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.

```
#include <stdio.h>

#define MAX_STUDENTS 5

// Define the structure for student information
struct Student
{
    char name[50];
    int rollNumber;
    float marks[3];
    float average;
};

// FUnction prototype
float calculateAverage(float marks[], int numSubjects);

int main()
{
    struct Student students[MAX_STUDENTS];
    int i, j;
```

```
    for (i = 0; i < MAX_STUDENTS; i++)
    {
        printf("Enter the details for student %d\n", i + 1);
        printf("Name: ");
        scanf(" %[^\n]s", students[i].name);
        printf("Roll Number: ");
        scanf("%d", &students[i].rollNumber);
        printf("Enter marks in 3 subjects\n");
        for (j = 0; j < 3; j++)
        {
            printf("Subject %d: ", j + 1);
            scanf("%f", &students[i].marks[j]);
        }

        // Call the function to calculate average marks
        students[i].average = calculateAverage(students[i].marks, 3);
    }

    printf("\nStudent Details\n");
    for (i = 0; i < MAX_STUDENTS; i++)
    {
        printf("Name: %s\n", students[i].name);
        printf("Roll Number: %d\n", students[i].rollNumber);
        printf("Marks: %.2f, %.2f, %.2f\n", students[i].marks[0],
students[i].marks[1], students[i].marks[2]);
        printf("Average Marks: %.2f\n", students[i].average);
        printf("\n");
    }
```

```
    return 0;
}

// Function to calculate the average marks for a student
float calculateAverage(float marks[], int numSubjects)
{
    float total = 0.0;
    for (int i = 0; i < numSubjects; i++)
        total += marks[i];
    return total / numSubjects;
}
```

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.

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

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

```
    char department[30];
};

// Function prototype
void displayHighSalary(struct Employee employees[], int count, float threshold);

int main()
{
    int numEmployees;
    float salaryThreshold;

    printf("Enter the number of employees: ");
    scanf("%d", &numEmployees);

    struct Employee employees[numEmployees];

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

```
    }

    printf("\nEnter the salary threshold: ");
    scanf("%f", &salaryThreshold);

    // Call the function to display employees with salary above the threshold
    displayHighSalary(employees, numEmployees, salaryThreshold);

    return 0;
}

// Function to display employees with salary above a threshold
void displayHighSalary(struct Employee employees[], int count, float threshold)
{
    printf("\nEmployees with salary above %.2f\n", threshold);
    int found = 0;
    for (int i = 0; i < count; i++)
    {
        if (employees[i].salary > threshold)
        {
            printf("Name: %s\n", employees[i].name);
            printf("ID: %d\n", employees[i].id);
            printf("Salary: %.2f\n", employees[i].salary);
            printf("Department: %s\n", employees[i].department);
            printf("\n");
            found = 1;
```

```
        }
    }
    if (!found)
        printf("No employees found with salary above %.2f\n", 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.

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

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

// Function prototype
void searchBookByTitle(struct Book books[], int count, char searchTitle[]);
```

```
int main()
{
    int numBooks;
    char searchTitle[100];

    printf("Enter the number of books in the inventory: ");
    scanf("%d", &numBooks);

    struct Book books[numBooks];

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

    printf("\nEnter the title of the book to search for: ");
    scanf(" %[^\n]s", searchTitle);

    // Call the function to search for the book by title
```

```
    searchBookByTitle(books, numBooks, searchTitle);
    return 0;
}

// Function to search for a book by title
void searchBookByTitle(struct Book books[], int count, char searchTitle[])
{
    int found = 0;
    printf("\nSearching for book titled \"%s\"\n", searchTitle);
    for (int i = 0; i < count; i++)
    {
        if (strcasecmp(books[i].title, searchTitle) == 0)
        {
            printf("Book 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);
            printf("\n");
            found = 1;
            break;
        }
    }
    if (!found)
        printf("No book found with the title \"%s\"\n", searchTitle);
}
```

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).

```
#include <stdio.h>

// Define the structure for a date
struct Date
{
    int day;
    int month;
    int year;
};

// Function prototype
int isValidDate(struct Date date);
int isLeapYear(int year);

int main()
{
    struct Date date;

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

    if (isValidDate(date))
```

```
        printf("Date %d/%d/%d is valid\n", date.day, date.month, date.year);
    else
        printf("Date %d/%d/%d is invalid\n", date.day, date.month, date.year);

    return 0;
}

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

// Function to validate if a date is correct
int isValidDate(struct Date date)
{
    int daysInMonth[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

    // Adjust for February in a leap year
    if (isLeapYear(date.year))
        daysInMonth[1] = 29;

    // Check year, month, and day validity
    if (date.year < 1 || date.month < 1 || date.month > 12 || date.day < 1 || date.day > daysInMonth[date.month - 1])
        return 0;
```

```
    return 1;
}
```

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.

```
#include <stdio.h>

// Define the structure for a complex number
struct Complex
{
    float real;
    float imaginary;
};

// Function prototypes
struct Complex add(struct Complex c1, struct Complex c2);
struct Complex sub(struct Complex c1, struct Complex c2);
struct Complex mul(struct Complex c1, struct Complex c2);
void displayComplex(struct Complex c);

int main()
{
    struct Complex c1, c2, result;
```

```
    printf("Enter the real and imaginary parts of the first complex number: ");
    scanf("%f %f", &c1.real, &c1.imaginary);

    printf("Enter the real and imaginary parts of the second complex number: ");
    scanf("%f %f", &c2.real, &c2.imaginary);

    // Call the function to perform addition
    result = add(c1, c2);
    printf("\nAddition: ");
    displayComplex(result);

    // Call the function to perform subtraction
    result = sub(c1, c2);
    printf("Subtraction: ");
    displayComplex(result);

    // Call the function to perform multiplication
    result = mul(c1, c2);
    printf("Multiplication: ");
    displayComplex(result);
    return 0;
}

// Function to add two complex numbers
struct Complex add(struct Complex c1, struct Complex c2)
```

```
{
    struct Complex result;
    result.real = c1.real + c2.real;
    result.imaginary = c1.imaginary + c2.imaginary;
    return result;
}

// Function to subtract two complex numbers
struct Complex sub(struct Complex c1, struct Complex c2)
{
    struct Complex result;
    result.real = c1.real - c2.real;
    result.imaginary = c1.imaginary - c2.imaginary;
    return result;
}

// Function to multiply two complex numbers
struct Complex mul(struct Complex c1, struct Complex c2)
{
    struct Complex result;
    result.real = c1.real * c2.real - c1.imaginary * c2.imaginary;
     result.imaginary = c1.real * c2.imaginary + c1.imaginary * c2.real;
    return result;
}

// Function to display a complex number
void displayComplex(struct Complex c)
```

```
{
    printf("%.2f + %.2fi\n", c.real, c.imaginary);
}
```

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.

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

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

// Function prototypes
void deposit(struct BankAccount *account, float amount);
void withdraw(struct BankAccount *account, float amount);
void displayAccount(struct BankAccount account);

int main()
```

```
{
    struct BankAccount account;

    printf("Enter account number: ");
    scanf("%d", &account.accountNumber);
    printf("Enter account holder name: ");
    scanf(" %[^\n]s", account.accountHolderName);
    printf("Enter initial balance: ");
    scanf("%f", &account.balance);

    int option;
    float amount;
    do
    {
        printf("Bank account menu:\n");
        printf("1. Deposit Money\n");
        printf("2. Withdraw Money\n");
        printf("3. Display account details\n");
        printf("4. Exit\n");
        printf("Enter the option: ");
        scanf("%d", &option);

        switch (option)
        {
            case 1: printf("Enter amount to deposit: ");
                    scanf("%f", &amount);
                    deposit(&account, amount);
```

```
            break;
        case 2: printf("Enter amount to withdraw: ");
            scanf("%f", &amount);
            withdraw(&account, amount);
            break;
        case 3: displayAccount(account);
            break;
        case 4: printf("Exit\n");
            break;
        default:printf("Invalid option\n");
      }
   } while (option != 4);
   return 0;
}

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

// Function to display account details
void displayAccount(struct BankAccount account)
{
    printf("\nAccount details\n");
    printf("Account number: %d\n", account.accountNumber);
    printf("Account holder: %s\n", account.accountHolderName);
    printf("Balance: %.2f\n", account.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.

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

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

// Function prototype
void displayCarsInPriceRange(struct Car cars[], int count,float minPrice, float maxPrice);

int main()
{
    int numCars;
    float maxPrice, minPrice;

    printf("Enter the number of cars in the inventory: ");
    scanf("%d", &numCars);

    struct Car cars[numCars];
```

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

    printf("\nEnter the minimum price: ");
    scanf("%f", &minPrice);
    printf("Enter the maximum price: ");
    scanf("%f", &maxPrice);

    // Call the function to display cars within the specified price range
    displayCarsInPriceRange(cars, numCars, minPrice, maxPrice);
    return 0;
}

// Function to display cars within a specified price range
void displayCarsInPriceRange(struct Car cars[], int count, float minPrice, float maxPrice)
{
```

```
    printf("\nCars within the price range %.2f - %.2f:\n", minPrice, maxPrice);
    int found = 0;
    for (int i = 0; i < count; i++)
    {
        if (cars[i].price >= minPrice && cars[i].price <= maxPrice)
        {
            printf("\nMake: %s\n", cars[i].make);
            printf("Model: %s\n", cars[i].model);
            printf("Year: %d\n", cars[i].year);
            printf("Price: %.2f\n", cars[i].price);
            found = 1;
        }
    }
    if (!found)
        printf("No cars found within the specified price range\n");
}
```

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.

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

// Define the structure for a library book
struct Book
```

```
{
    char title[100];
    char author[100];
    int publicationYear;
    char status[10];
};

// Function prototypes
void issueBook(struct Book *book);
void returnBook(struct Book *book);
void displayBooks(struct Book books[], int count);

int main()
{
    int numBooks, option, bookIndex, count;

    printf("Enter the number of books in the library: ");
    scanf("%d", &numBooks);

    struct Book books[numBooks];

    for (int i = 0; i < numBooks; i++)
    {
        printf("\nEnter details for book %d:\n", i + 1);
        printf("Title: ");
        scanf(" %[^\n]s", books[i].title);
        printf("Author: ");
```

```
        scanf(" %[^\n]s", books[i].author);
        printf("Publication Year: ");
        scanf("%d", &books[i].publicationYear);
        strcpy(books[i].status, "available");
    }
    do
    {
        printf("\nLibrary management menu:\n");
        printf("1. Issue a Book\n");
        printf("2. Return a Book\n");
        printf("3. Display All Books\n");
        printf("4. Exit\n");
        printf("Enter the option: ");
        scanf("%d", &option);

        switch (option)
        {
            case 1: printf("Enter the index of the book to issue (1 to %d):
", numBooks);
                    scanf("%d", &bookIndex);
                    if (bookIndex >= 1 && bookIndex <= numBooks)
                        issueBook(&books[bookIndex - 1]);
                    else
                        printf("Invalid book index\n");
                    break;
           case 2: printf("Enter the index of the book to return (1 to %d):
", numBooks);
                    scanf("%d", &bookIndex);
```

```
                if (bookIndex >= 1 && bookIndex <= numBooks)
                    returnBook(&books[bookIndex - 1]);
                else
                    printf("Invalid book index\n");
                break;
            case 3: displayBooks(books, numBooks);
                break;
            case 4: printf("Exit\n");
                break;
            default:printf("Invalid option\n");
        }
    } while (option != 4);
    return 0;
}

// Function to issue a book
void issueBook(struct Book *book)
{
    if (strcmp(book->status, "available") == 0)
    {
        strcpy(book->status, "issued");
        printf("The book \"%s\" has been successfully issued\n", book->title);
    }
    else
        printf("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("The book \"%s\" has been successfully returned\n", book->title);
    }
    else
        printf("The book \"%s\" is not returned\n", book->title);
}

// Function to display details of all books
void displayBooks(struct Book books[], int count) {
    printf("\nLibrary Books:\n");
    for (int i = 0; i < count; i++) {
        printf("\nTitle: %s\n", books[i].title);
        printf("Author: %s\n", books[i].author);
        printf("Publication Year: %d\n", books[i].publicationYear);
        printf("Status: %s\n", books[i].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.

```
#include <stdio.h>

#define MAX_GRADES 3

// Structure to store student information
struct Student
{
    char name[50];
    int roll_no;
    int grades[MAX_GRADES];
};

// Function prototypes
float calculateAverage(int grades[], int size);
int findHighestGrade(int grades[], int size);
int findLowestGrade(int grades[], int size);

int main()
{
    int numStudents;

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

    struct Student students[numStudents];
```

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

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

        printf("Enter roll number: ");
        scanf("%d", &students[i].roll_no);

        printf("Enter grades for %d subjects:\n", MAX_GRADES);
        for (int j = 0; j < MAX_GRADES; j++)
        {
            printf("Grade %d: ", j + 1);
            scanf("%d", &students[i].grades[j]);
        }
    }

    for (int i = 0; i < numStudents; i++)
    {
            printf("\nStudent %d: %s (Roll No: %d)\n", i + 1,
students[i].name, students[i].roll_no);
        printf("Grades: ");
        for (int j = 0; j < MAX_GRADES; j++)
            printf("%d ", students[i].grades[j]);
```

```
            int highest = findHighestGrade(students[i].grades,
MAX_GRADES);
            int lowest = findLowestGrade(students[i].grades,
MAX_GRADES);
            float average = calculateAverage(students[i].grades,
MAX_GRADES);

        printf("\nHighest Grade: %d\n", highest);
        printf("Lowest Grade: %d\n", lowest);
        printf("Average Grade: %.2f\n", average);
    }
    return 0;
}

// Function to calculate the average grade
float calculateAverage(int grades[], int size)
{
    int sum = 0;
    for (int i = 0; i < size; i++)
        sum += grades[i];
    return (float)sum / size;
}

// Function to find the highest grade
int findHighestGrade(int grades[], int size)
{
    int highest = grades[0];
    for (int i = 1; i < size; i++)
```

```
    {
        if (grades[i] > highest)
            highest = grades[i];
    }
    return highest;
}

// Function to find the lowest grade
int findLowestGrade(int grades[], int size)
{
    int lowest = grades[0];
    for (int i = 1; i < size; i++)
    {
        if (grades[i] < lowest)
            lowest = grades[i];
    }
    return lowest;
}
```

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 MAX_PRODUCTS 3

// Structure to store product information
struct Product
{
    int product_id;
    char name[50];
    int quantity;
    float price;
};

// Function prototypes
void updateQuantity(struct Product* product, int soldQuantity);
float calculateSalesValue(struct Product* product, int soldQuantity);

int main()
{
    int numProducts = MAX_PRODUCTS;
    float totalSalesValue = 0.0;

    struct Product products[MAX_PRODUCTS] = {
        {101, "Headphones", 200, 50.00},
        {102, "Monitor", 30, 200.00},
        {103, "Keyboard", 150, 40.00}
    };

    printf("Product catalog:\n");
```

```
    printf("ID\tName\t\tQuantity\tPrice\n");
    for (int i = 0; i < numProducts; i++)
        printf("%d\t%s\t%d\t\t%.2f\n", products[i].product_id, products[i].name, products[i].quantity, products[i].price);

    for (int i = 0; i < numProducts; i++)
    {
        int soldQuantity;
        printf("\nEnter the quantity sold for %s: ", products[i].name);
        scanf("%d", &soldQuantity);

        // Call the function to update product quantity and calculate sales value
        updateQuantity(&products[i], soldQuantity);
        totalSalesValue += calculateSalesValue(&products[i], soldQuantity);
    }
    printf("\nTotal Sales Value: %.2f\n", totalSalesValue);
    return 0;
}

// Function to update product quantity after a sale
void updateQuantity(struct Product* product, int soldQuantity)
{
    if (soldQuantity <= product->quantity)
    {
        product->quantity -= soldQuantity;
       printf("Sale successful, updated quantity of %s: %d\n", product->name, product->quantity);
```

```
    }
    else
        printf("Not enough stock for product %s, available quantity: %d\n", product->name, product->quantity);
}

// Function to calculate the total sales value for a given product
float calculateSalesValue(struct Product* product, int soldQuantity)
{
    if (soldQuantity <= product->quantity)
        return soldQuantity * product->price;
    else
    {
        printf("Not enough stock to calculate sales value for %s\n", product->name);
        return 0;
    }
}
```

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.

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

// Structure to represent a point in 2D space
struct Point
{
   float x;
   float y;
};

//Function prototype
float calculateDistance(struct Point p1, struct Point p2);

int main()
{
   struct Point point1, point2;

   printf("Enter coordinates of point 1 (x1 y1): ");
   scanf("%f %f", &point1.x, &point1.y);

   printf("Enter coordinates of point 2 (x2 y2): ");
   scanf("%f %f", &point2.x, &point2.y);
```

```
    // Call the function to calculate and display the distance
    float distance = calculateDistance(point1, point2);
    printf("The distance between the points (%.2f, %.2f) and (%.2f, %.2f) is: %.2f\n",
           point1.x, point1.y, point2.x, point2.y, distance);
    return 0;
}

// Function to calculate the distance between two points
float calculateDistance(struct Point p1, struct Point p2)
{
    // Distance formula: sqrt((x2 - x1)^2 + (y2 - y1)^2)
    return sqrt(pow(p2.x - p1.x, 2) + pow(p2.y - p1.y, 2));
}
```

2. Rectangle Properties:
    - Create a structure for a rectangle with length and width.
    - Write functions to calculate the area and perimeter of the rectangle.

```
#include <stdio.h>

// Structure to represent a rectangle
struct Rectangle
{
    float length;
    float width;
};
```

```
//Function prototypes
float calculateArea(struct Rectangle rect);
float calculatePerimeter(struct Rectangle rect);

int main()
{
   struct Rectangle rect;

   printf("Enter the length and width of the rectangle: ");
   scanf("%f %f", &rect.length, &rect.width);

   // Call the function to calculate and display the area and perimeter
   float area = calculateArea(rect);
   float perimeter = calculatePerimeter(rect);

   printf("Area of the rectangle: %.2f\n", area);
   printf("Perimeter of the rectangle: %.2f\n", perimeter);
   return 0;
}

// Function to calculate the area of the rectangle
float calculateArea(struct Rectangle rect)
{
   return rect.length * rect.width;
}

// Function to calculate the perimeter of the rectangle
```

```
float calculatePerimeter(struct Rectangle rect)
{
   return 2 * (rect.length + rect.width);
}
```

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.

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

// Structure to store movie details
struct Movie
{
   char title[100];
   char director[100];
   int releaseYear;
   float rating;
};

// Function prototype
int compareMovies(const void *a, const void *b);

int main()
{
```

```
    int numMovies;
    printf("Enter the number of movies: ");
    scanf("%d", &numMovies);

    struct Movie movies[numMovies];

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

        printf("Enter title: ");
        scanf(" %[^\n]", movies[i].title);

        printf("Enter director: ");
        scanf(" %[^\n]", movies[i].director);

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

        printf("Enter rating: ");
        scanf("%f", &movies[i].rating);
    }

    // Sort the movies based on their rating
        qsort(movies, numMovies, sizeof(struct Movie), compareMovies);

    printf("\nMovies sorted by rating\n");
```

```
    for (int i = 0; i < numMovies; i++)
    {
        printf("\nTitle: %s\n", movies[i].title);
        printf("Director: %s\n", movies[i].director);
        printf("Release Year: %d\n", movies[i].releaseYear);
        printf("Rating: %.2f\n", movies[i].rating);
    }
    return 0;
}

// Function to compare two movies based on their rating (for sorting)
int compareMovies(const void *a, const void *b)
{
    struct Movie *movieA = (struct Movie *)a;
    struct Movie *movieB = (struct Movie *)b;
    if (movieA->rating < movieB->rating)
        return -1;
    else if (movieA->rating > movieB->rating)
        return 1;
    return 0;
}
```

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.

```
#include <stdio.h>

// Structure to store daily weather data
struct Weather
{
    char date[11];
    float temperature;
    float humidity;
};

// Function prototype
int findHighestTemperatureDay(struct Weather data[], int numDays);

int main()
{
    int numDays;

    printf("Enter the number of days: ");
    scanf("%d", &numDays);

    struct Weather weatherData[numDays];

    for (int i = 0; i < numDays; i++)
    {
        printf("\nEnter data for day %d\n", i + 1);

        printf("Enter date (YYYY-MM-DD): ");
```

```
        scanf("%s", weatherData[i].date);

        printf("Enter temperature: ");
        scanf("%f", &weatherData[i].temperature);

        printf("Enter humidity: ");
        scanf("%f", &weatherData[i].humidity);
    }

    // Call the function to find the day with the highest temperature
 int    highestTempDayIndex    =
        findHighestTemperatureDay(weatherData, numDays);
      printf("\nThe day with the highest temperature is: %s\n",
weatherData[highestTempDayIndex].date);
         printf("Temperature:%.2f°C\n",weatherData[highestTempD
ayIndex].temperature);
        printf("Humidity:%.2f%%\n",weatherData[highestTempDay
Index].humidity);
    return 0;
}

// Function to find the day with the highest temperature
int findHighestTemperatureDay(struct Weather data[], int numDays)
{
    int highestTempDayIndex = 0;
    for (int i = 1; i < numDays; i++)
    {
        if      (data[i].temperature        >
        data[highestTempDayIndex].temperature)
```

```
            highestTempDayIndex = i;
        }
        return highestTempDayIndex;
    }
```

5. Fraction Arithmetic:
    - Define a structure for a fraction with numerator and denominator.
    - Write functions to add, subtract, multiply, and divide two fractions.

```
#include <stdio.h>

// Structure to represent a fraction
struct Fraction
{
    int numerator;
    int denominator;
};

// Function prototype
int gcd(int a, int b);
void simplify(struct Fraction *frac);
struct Fraction add(struct Fraction f1, struct Fraction f2);
struct Fraction sub(struct Fraction f1, struct Fraction f2);
struct Fraction mul(struct Fraction f1, struct Fraction f2);
struct Fraction div(struct Fraction f1, struct Fraction f2);
void display(struct Fraction f);
```

```
int main()
{
    struct Fraction f1, f2, result;

    printf("Enter the numerator and denominator for the first fraction: ");
    scanf("%d %d", &f1.numerator, &f1.denominator);

    printf("Enter the numerator and denominator for the second fraction: ");
    scanf("%d %d", &f2.numerator, &f2.denominator);

    // Call the function to add fractions
    result = add(f1, f2);
    printf("Sum: ");
    display(result);
    printf("\n");

    // Call the function to subtract fractions
    result = sub(f1, f2);
    printf("Difference: ");
    display(result);
    printf("\n");

    // Multiply fractions
    result = mul(f1, f2);
    printf("Product: ");
    display(result);
```

```
    printf("\n");

    // Divide fractions
    result = div(f1, f2);
    printf("Quotient: ");
    display(result);
    printf("\n");
    return 0;
}

// Function to find the Greatest Common Divisor (GCD) of two numbers
int gcd(int a, int b)
{
    while (b != 0)
    {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}

// Function to simplify the fraction
void simplify(struct Fraction *frac)
{
    int commonDivisor = gcd(frac->numerator, frac->denominator);
    frac->numerator /= commonDivisor;
```

```
    frac->denominator /= commonDivisor;
}

// Function to add two fractions
struct Fraction add(struct Fraction f1, struct Fraction f2)
{
    struct Fraction result;
    result.numerator = f1.numerator * f2.denominator + f2.numerator
* f1.denominator;
    result.denominator = f1.denominator * f2.denominator;
    simplify(&result);
    return result;
}

// Function to subtract two fractions
struct Fraction sub(struct Fraction f1, struct Fraction f2)
{
    struct Fraction result;
    result.numerator = f1.numerator * f2.denominator - f2.numerator
* f1.denominator;
    result.denominator = f1.denominator * f2.denominator;
    simplify(&result);
    return result;
}

// Function to multiply two fractions
struct Fraction mul(struct Fraction f1, struct Fraction f2)
{
```

```
    struct Fraction result;
    result.numerator = f1.numerator * f2.numerator;
    result.denominator = f1.denominator * f2.denominator;
    simplify(&result);
    return result;
}

// Function to divide two fractions
struct Fraction div(struct Fraction f1, struct Fraction f2)
{
    struct Fraction result;
    result.numerator = f1.numerator * f2.denominator;
    result.denominator = f1.denominator * f2.numerator;
    simplify(&result);
    return result;
}

// Function to display a fraction
void display(struct Fraction f)
{
    printf("%d/%d", f.numerator, f.denominator);
}
```

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.

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

// Structure to represent a laptop
struct Laptop
{
    char brand[50];
    char model[50];
    char processor[50];
    int RAM;
    float price;
};

// Function prototype
void displayLaptop(struct Laptop laptop);

int main()
{
    int numLaptops, i;
    float minPrice, maxPrice;

    printf("Enter the number of laptops: ");
    scanf("%d", &numLaptops);

    struct Laptop laptops[numLaptops];
```

```
    for (i = 0; i < numLaptops; i++)
    {
        printf("\nEnter details for laptop %d\n", i + 1);

        printf("Enter brand: ");
        scanf(" %[^\n]", laptops[i].brand);

        printf("Enter model: ");
        scanf(" %[^\n]", laptops[i].model);

        printf("Enter processor: ");
        scanf(" %[^\n]", laptops[i].processor);

        printf("Enter RAM: ");
        scanf("%d", &laptops[i].RAM);

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

    printf("\nEnter the minimum price: ");
    scanf("%f", &minPrice);
    printf("Enter the maximum price: ");
    scanf("%f", &maxPrice);

    printf("\nLaptops within the price range %.2f - %.2f:\n", minPrice,
maxPrice);
    printf("------------------------\n");
```

```
    int found = 0;
    for (i = 0; i < numLaptops; i++)
    {
            if (laptops[i].price >= minPrice && laptops[i].price <= maxPrice)
        {
            displayLaptop(laptops[i]);
            found = 1;
        }
    }
    if (!found)
        printf("No laptops found within the specified price range\n");
    return 0;
}

// Function to display laptop details
void displayLaptop(struct 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);
    printf("-----------------------\n");
}
```

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.

```
#include <stdio.h>

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

void calculateAttendance(struct Attendance students[], int n);

int main()
{
    struct Attendance students[] = {{101, 50, 45}, {102, 50, 50}, {103, 50, 30}};
   int n = sizeof(students) / sizeof(students[0]);
   calculateAttendance(students, n);
   return 0;
}

void calculateAttendance(struct Attendance students[], int n)
{
```

```
    for (int i = 0; i < n; i++)
    {
        float percentage = (float)students[i].classesAttended / students[i].totalClasses * 100;
        printf("Student ID: %d, Attendance Percentage: %.2f%%\n", students[i].studentID, percentage);
    }
}
```

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.

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

struct Flight
{
    char flightNumber[10];
    char departure[20];
    char destination[20];
    float duration;
} ;

void displayFlights(struct Flight flights[], int n, float maxDuration);
```

```
int main()
{
   struct Flight flights[] = {
      {"AI101", "Delhi", "Mumbai", 2.0},
      {"AI102", "Delhi", "Bangalore", 2.5},
      {"AI103", "Mumbai", "Chennai", 1.0}
   };
   int n = sizeof(flights) / sizeof(flights[0]);
   displayFlights(flights, n, 1.5);
   return 0;
}

void displayFlights(struct Flight flights[], int n, float maxDuration)
{
   for (int i = 0; i < n; i++)
   {
      if (flights[i].duration < maxDuration)
      {
            printf("Flight: %s, From: %s, To: %s, Duration: %.2f hours\n",
                  flights[i].flightNumber, flights[i].departure, flights[i].destination, flights[i].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.

```
#include <stdio.h>

struct PolynomialTerm
{
   int coefficient;
   int exponent;
};

void add(struct PolynomialTerm p1[], int n1, struct PolynomialTerm p2[], int n2);
void mul(struct PolynomialTerm p1[], int n1, struct PolynomialTerm p2[], int n2);

int main()
{
   struct PolynomialTerm poly1[] = {{3, 2}, {4, 1}};
   struct PolynomialTerm poly2[] = {{2, 1}, {1, 0}};
   add(poly1, 2, poly2, 2);
   mul(poly1, 2, poly2, 2);
   return 0;
}

void add(struct PolynomialTerm p1[], int n1, struct PolynomialTerm p2[], int n2)
```

```
{
    printf("Addition:\n");
    for (int i = 0; i < n1; i++)
    {
            printf("%dx^%d ", p1[i].coefficient + p2[i].coefficient,
p1[i].exponent);
        if (i < n1 - 1)
            printf("+ ");
    }
    printf("\n");
}

void mul(struct PolynomialTerm p1[], int n1, struct PolynomialTerm
p2[], int n2)
{
    printf("Multiplication:\n");
    for (int i = 0; i < n1; i++)
    {
        for (int j = 0; j < n2; j++)
        {
            printf("%dx^%d ", p1[i].coefficient * p2[j].coefficient,
                p1[i].exponent + p2[j].exponent);
            if (i < n1 - 1 || j < n2 - 1)
                printf("+ ");
        }
    }
    printf("\n");
}
```

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.

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

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

void searchByDiagnosis(struct MedicalRecord records[], int n, char diagnosis[]);

int main()
{
   struct MedicalRecord records[] = {
      {"Lily", 25, "Flu", "Rest and hydration"},
      {"Vandana", 40, "Diabetes", "Insulin therapy"},
      {"Ananya", 35, "Flu", "Medication and rest"}
   };
   int n = sizeof(records) / sizeof(records[0]);
   searchByDiagnosis(records, n, "Flu");
```

```
    return 0;
}

void searchByDiagnosis(struct MedicalRecord records[], int n, char
diagnosis[])
{
    for (int i = 0; i < n; 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);
        }
    }
}
```

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.

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

struct Player
{
    char name[50];
```

```
    char game[50];
    int score;
};

void displayTopScorer(struct Player players[], int n);

int main()
{
    struct Player players[] = {
        {"Hemanth", "Chess", 80},
        {"Shashank", "Chess", 70},
        {"Lanky", "Tennis", 65},
        {"Veeresh", "Tennis", 90},
        {"Anushik", "Football", 75}
    };
    int n = sizeof(players) / sizeof(players[0]);
    displayTopScorer(players, n);
    return 0;
}

void displayTopScorer(struct Player players[], int n)
{
    char currentGame[50] = "";
    int maxScore = 0;
    char topScorer[50] = "";

    for (int i = 0; i < n; i++) {
```

```
        if (strcmp(currentGame, players[i].game) != 0)
        {
            if (i > 0)
                printf("Game: %s, Top Scorer: %s, Score: %d\n",
currentGame, topScorer, maxScore);
            strcpy(currentGame, players[i].game);
            maxScore = players[i].score;
            strcpy(topScorer, players[i].name);
        }
        else if (players[i].score > maxScore)
        {
            maxScore = players[i].score;
            strcpy(topScorer, players[i].name);
        }
    }
    printf("Game: %s, Top Scorer: %s, Score: %d\n", currentGame,
topScorer, maxScore);
}
```

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.

```
#include <stdio.h>
struct City
{
```

```
    char name[50];
    int population;
    float area;
} ;

void displayPopulationDensity(struct City cities[], int n);

int main()
{
    struct City cities[] = {
        {"City A", 1000000, 700.8},
        {"City B", 2000000, 950.2},
        {"City C", 500000, 500.5}
    };
    int n = sizeof(cities) / sizeof(cities[0]);
    displayPopulationDensity(cities, n);
    return 0;
}

void displayPopulationDensity(struct City cities[], int n)
{
    for (int i = 0; i < n; i++)
    {
        float density = cities[i].population / cities[i].area;
         printf("City: %s, Population Density: %.2f people/sq.km\n",
cities[i].name, density);
    }
}
```

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.

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

struct Vehicle
{
   char regNumber[20];
   char owner[50];
   char make[50];
   int year;
};

void listVehiclesByYear(struct Vehicle vehicles[], int n, int year);

int main()
{
   struct Vehicle vehicles[] = {
      {"KA18AB1234", "Arjun", "Mahindra", 2022},
      {"KA02CD5678", "Hansika", "Honda", 2021},
      {"KA03EF9101", "Dhruthi", "Ford", 2020}
   };
   int n = sizeof(vehicles) / sizeof(vehicles[0]);
   listVehiclesByYear(vehicles, n, 2020);
   return 0;
```

```
}

void listVehiclesByYear(struct Vehicle vehicles[], int n, int year)
{
   for (int i = 0; i < n; i++)
   {
      if (vehicles[i].year == year)
      {
         printf("Reg. No: %s, Owner: %s, Make: %s\n",
         vehicles[i].regNumber, vehicles[i].owner, vehicles[i].make);
      }
   }
}
```

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.

```
#include <stdio.h>
#include <string.h>
struct MenuItem
{
   char name[50];
   char category[50];
   float price;
};
```

```
void displayItemsByCategory(struct MenuItem menu[], int n, char category[]);


int main()
{
   struct MenuItem menu[] = {
      {"Pizza", "Main Course", 8.5},
      {"Pasta", "Main Course", 7.9},
      {"Cake", "Dessert", 6.5},
      {"Ice Cream", "Dessert", 4.99}
   };
   int n = sizeof(menu) / sizeof(menu[0]);
   displayItemsByCategory(menu, n, "Dessert");
   return 0;
}


void displayItemsByCategory(struct MenuItem menu[], int n, char category[])
{
   for (int i = 0; i < n; i++)
   {
      if (strcmp(menu[i].category, category) == 0)
                  printf("Name: %s, Price: %.2f\n", menu[i].name, menu[i].price);
   }
}
```

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.

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

struct Team
{
   char teamName[50];
   char sport[50];
   int numberOfPlayers;
   char coach[50];
};

void displayTeamsBySport(struct Team teams[], int n, char sport[]);

int main()
{
   struct Team teams[] = {
      {"Lions", "Football", 11, "Diganth"},
      {"Tigers", "Basketball", 5, "Aravind"},
      {"Sharks", "Football", 11, "Raghu"}
   };
   int n = sizeof(teams) / sizeof(teams[0]);
   displayTeamsBySport(teams, n, "Football");
   return 0;
```

```
}

void displayTeamsBySport(struct Team teams[], int n, char sport[])
{
   for (int i = 0; i < n; i++)
   {
      if (strcmp(teams[i].sport, sport) == 0)
         printf("Team Name: %s, Coach: %s, Players: %d\n",
   teams[i].teamName, teams[i].coach, teams[i].numberOfPlayers);
   }
}
```

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.

```
#include <stdio.h>

struct Student
{
   char name[50];
   int marks[5];
};

void calculateResults(struct Student students[], int n);

int main()
```

```
{
    struct Student students[] = {
        {"Varun", {80, 85, 90, 75, 95}},
        {"Divya", {70, 65, 75, 80, 85}},
        {"Anjali", {60, 70, 80, 90, 100}}
    };
    int n = sizeof(students) / sizeof(students[0]);
    calculateResults(students, n);
    return 0;
}

void calculateResults(struct Student students[], int n)
{
    for (int i = 0; i < n; i++)
    {
        int total = 0;
        for (int j = 0; j < 5; j++)
            total += students[i].marks[j];
        float percentage = (float)total / 5;
            printf("Name: %s, Total: %d, Percentage: %.2f%%\n",
students[i].name, total, percentage);
    }
}
```

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.

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

struct Product
{
    int productID;
    char name[50];
    char category[50];
    float price;
    int stock;
};

void updateStock(struct Product products[], int n, int productID, int quantity);
void calculateTotalValue(struct Product products[], int n);

int main()
{
    struct Product products[] = {
        {101, "Laptop", "Electronics", 999.99, 10},
        {102, "Phone", "Electronics", 699.99, 15},
        {103, "Headphones", "Accessories", 199.99, 20}
    };
    int n = sizeof(products) / sizeof(products[0]);
    updateStock(products, n, 101, 5);
```

```
    calculateTotalValue(products, n);
    return 0;
}

void updateStock(struct Product products[], int n, int productID, int quantity)
{
    for (int i = 0; i < n; i++)
    {
        if (products[i].productID == productID)
        {
            products[i].stock += quantity;
                printf("Updated stock for %s: %d\n", products[i].name, products[i].stock);
            return;
        }
    }
    printf("Product not found.\n");
}

void calculateTotalValue(struct Product products[], int n)
{
    float totalValue = 0;
    for (int i = 0; i < n; i++)
        totalValue += products[i].price * products[i].stock;
    printf("Total value of products in stock: %.2f\n", totalValue);
}
```

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.

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

struct Album
{
   char albumName[50];
   char artist[50];
   char genre[50];
   int releaseYear;
};

void displayAlbumsByGenre(struct Album albums[], int n, char genre[]);

int main()
{
   struct Album albums[] = {
      {"Album A", "Artist X", "Rock", 2010},
      {"Album B", "Artist Y", "Pop", 2015},
      {"Album C", "Artist Z", "Rock", 2020}
   };
   int n = sizeof(albums) / sizeof(albums[0]);
   displayAlbumsByGenre(albums, n, "Rock");
```

```
    return 0;
}

void displayAlbumsByGenre(struct Album albums[], int n, char genre[])
{
    for (int i = 0; i < n; i++)
    {
        if (strcmp(albums[i].genre, genre) == 0)
            printf("Album: %s, Artist: %s, Year: %d\n",
       albums[i].albumName, albums[i].artist, albums[i].releaseYear);
    }
}
```

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.

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

struct Ticket
{
    char movieName[50];
    int seatNumber;
```

```
    float price;
};

void bookTickets(struct Ticket tickets[], int *n, char movieName[], int seatNumber, float price);
float calculateTotalRevenue(struct Ticket tickets[], int n);

int main()
{
    struct Ticket tickets[100];
    int n = 0;

    bookTickets(tickets, &n, "Movie A", 1, 10.0);
    bookTickets(tickets, &n, "Movie A", 2, 10.0);
    bookTickets(tickets, &n, "Movie B", 1, 12.0);

     printf("Total Revenue: %.2f\n", calculateTotalRevenue(tickets, n));
    return 0;
}

void bookTickets(struct Ticket tickets[], int *n, char movieName[], int seatNumber, float price)
{
    strcpy(tickets[*n].movieName, movieName);
    tickets[*n].seatNumber = seatNumber;
    tickets[*n].price = price;
    (*n)++;
```

```
}

float calculateTotalRevenue(struct Ticket tickets[], int n)
{
    float total = 0;
    for (int i = 0; i < n; i++)
        total += tickets[i].price;
    return total;
}
```

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 <string.h>

struct Course
{
    char courseCode[10];
    char name[50];
    char instructor[50];
    int credits;
};
```

```
void listCoursesByInstructor(struct Course courses[], int n, char instructor[]);

int main()
{
    struct Course courses[] = {
        {"CS101", "Intro to CS", "Dr. Smith", 3},
        {"CS102", "Data Structures", "Dr. Brown", 4},
        {"CS103", "Algorithms", "Dr. Smith", 4}
    };
    int n = sizeof(courses) / sizeof(courses[0]);
    listCoursesByInstructor(courses, n, "Dr. Smith");
    return 0;
}

void listCoursesByInstructor(struct Course courses[], int n, char instructor[])
{
    for (int i = 0; i < n; i++)
    {
        if (strcmp(courses[i].instructor, instructor) == 0)
            printf("Course Code: %s, Course Name: %s, Credits: %d\n",
                                courses[i].courseCode, courses[i].name, courses[i].credits);
    }
}
```