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>
struct Student {
   char name[50];
   int rollNumber;
   float marks[3];
};

int main() {
   struct Student students[5];
   for (int i = 0; i < 5; i++) {
      printf("Enter details for Student %d\n", i + 1);

      printf("Enter name: ");
      scanf("%s", students[i].name);
      printf("Enter roll number: ");
      scanf("%d", &students[i].rollNumber);
      printf("Enter marks for 3 subjects: ");
      for (int j = 0; j < 3; j++) {
         scanf("%f", &students[i].marks[j]);
      }
      printf("\n");
   }
   printf("\nStudent Details and Average Marks:\n");
   for (int i = 0; i < 5; i++) {
      float total = 0;
      for (int j = 0; j < 3; j++) {
         total += students[i].marks[j];
```

```
        }

        float average = total / 3;

        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\n", average);

    }

    return 0;

}
```

o/p:

```
Enter details for Student 1

Enter name: sofi

Enter roll number: 67

Enter marks for 3 subjects: 67 89 90


Enter details for Student 2

Enter name: sanjay

Enter roll number: 66

Enter marks for 3 subjects: 78 90 97


Enter details for Student 3

Enter name: christo

Enter roll number: 32

Enter marks for 3 subjects: 56 43 78


Enter details for Student 4

Enter name: arul

Enter roll number: 12

Enter marks for 3 subjects: 23 45 89
```

Enter details for Student 5

Enter name: lilly

Enter roll number: 34

Enter marks for 3 subjects: 98 76 54

Student Details and Average Marks:

Name: sofi

Roll Number: 67

Marks: 67.00, 89.00, 90.00

Average Marks: 82.00

Name: sanjay

Roll Number: 66

Marks: 78.00, 90.00, 97.00

Average Marks: 88.33

Name: christo

Roll Number: 32

Marks: 56.00, 43.00, 78.00

Average Marks: 59.00

Name: arul

Roll Number: 12

Marks: 23.00, 45.00, 89.00

Average Marks: 52.33

Name: lilly

Roll Number: 34

Marks: 98.00, 76.00, 54.00

Average Marks: 76.00

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>
struct Employee {
   char name[50];
   int id;
   float salary;
};
int main() {
   struct Employee employees[3] = {
      {"sofia", 1, 50000},
      {"Mickey", 2, 70000},
      {"Mickelen", 3, 40000}
   };
   float threshold = 60000;
   for (int i = 0; i < 3; i++) {
      if (employees[i].salary > threshold) {
         printf("\nName:%s\nID:%d\nSalary:%.2f\n", employees[i].name, employees[i].id, employees[i].salary);
      }
   }
   return 0;
}
```

o/p:

Name:Mickey

ID:2

Salary:70000.00

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>
struct Book {
   char title[50];
   char author[50];
   float price;
};

int main() {
   struct Book books[3] = {
      {"C", "Dennis", 25.50},
      {"jungle", "Kipling", 30.00},
      {"Harry", "Rowling", 35.75}
};

   char searchTitle[100];
   printf("Enter the title:");
   scanf("%s",searchTitle);
   for (int i = 0; i < 3; i++) {
      if (strcmp(books[i].title, searchTitle) == 0) {
         printf("Book:%s author: %s, Price: %.2f\n", books[i].title, books[i].author, books[i].price);
         return 0;
      }
   }

   printf("Book not found.\n");
   return 0;
}
```

o/p:

```
Enter the title:C
Book:C author: Dennis, Price: 25.50
```

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>
struct Date {
   int day;
   int month;
   int year;
};
int isValidDate(struct Date date) {
   if (date.month < 1 || date.month > 12)
   return 0;
   int daysInMonth[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
   if (date.year % 4 == 0) {
      daysInMonth[1] = 29;
   }
   return date.day >= 1 && date.day <= daysInMonth[date.month - 1];
}
int main() {
   struct Date date = {29, 2, 2023};
   if (isValidDate(date)) {
      printf("Date is valid.\n");
   } else {
      printf("Date is invalid.\n");
   }

   return 0;
}
```

o/p:

Date is invalid

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>

struct Complex {
    float real;
    float imaginary;
};

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

    printf("Enter first complex number: ");
    scanf("%f %f", &c1.real, &c1.imaginary);

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

    result.real = c1.real + c2.real;
    result.imaginary = c1.imaginary + c2.imaginary;
    printf("Addition: %.2f + %.2f\n", result.real, result.imaginary);

    result.real = c1.real - c2.real;
    result.imaginary = c1.imaginary - c2.imaginary;
    printf("Subtraction: %.2f + %.2f\n", result.real, result.imaginary);

    result.real = (c1.real * c2.real) - (c1.imaginary * c2.imaginary);
    result.imaginary = (c1.real * c2.imaginary) + (c1.imaginary * c2.real);
```

```
    printf("Multiplication:  %.2f + %.2f\n", result.real, result.imaginary);


    return 0;
}
```

o/p:

```
Enter first complex number: 3 4
Enter second complex number: 1 2
Addition: 4.00 + 6.00
Subtraction: 2.00 + 2.00
Multiplication:  -5.00 + 10.00
```

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>
struct BankAccount {
    int accountNumber;
    char holderName[50];
    float balance;
};

int main() {
    struct BankAccount account = {12345, "John Doe", 5000.0};
    float depositAmount, withdrawAmount;

    printf("Initial Balance: %.2f\n", account.balance);

    printf("Enter deposit amount: ");
    scanf("%f", &depositAmount);
    account.balance += depositAmount;
```

```
    printf("Enter withdrawal amount: ");
    scanf("%f", &withdrawAmount);
    if (account.balance >= withdrawAmount) {
        account.balance -= withdrawAmount;
    } else {
        printf("Insufficient balance.\n");
    }

    printf("Updated Balance: %.2f\n", account.balance);
    return 0;
}
```

o/p:

```
Initial Balance: 5000.00
Enter deposit amount: 4000
Enter withdrawal amount: 7000
Updated Balance: 2000.00
```

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>

struct Car {
    char make[50];
    int year;
    float price;
};

int main() {
    struct Car cars[3] = {
        {"Toyota", 2020, 20000},
```

```
    {"Honda", 2021, 25000},
    {"Ford", 2019, 15000}
  };

  float minPrice, maxPrice;
  printf("Enter min and max price: ");
  scanf("%f %f", &minPrice, &maxPrice);

  for (int i = 0; i < 3; i++) {
    if (cars[i].price >= minPrice && cars[i].price <= maxPrice) {
      printf("Car: %s, Year: %d, Price: %.2f\n", cars[i].make, cars[i].year, cars[i].price);
    }
  }

  return 0;
}
```

o/p:

```
Enter min and max price: 15000 25000
Car: Toyota, Year: 2020, Price: 20000.00
Car: Honda, Year: 2021, Price: 25000.00
Car: Ford, Year: 2019, Price: 15000.00
```

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>
struct Book {
  char title[50];
  char author[50];
  int publicationYear;
```

```
    char status[10];
};

// 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 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("The book '%s' has been returned.\n", book->title);
    } else {
        printf("The book '%s' was not issued.\n", book->title);
    }
}

int main() {
    struct Book books[3] = {
        {"The Great Gatsby", "F. Scott Fitzgerald", 1925, "available"},
        {"1984", "George Orwell", 1949, "available"},
        {"To Kill a Mockingbird", "Harper Lee", 1960, "available"}
    };
    printf("Issuing '1984':\n");
    issueBook(&books[1]);
```

```
    printf("\nReturning '1984':\n");
    returnBook(&books[1]);
    return 0;
}
```

o/p:

```
Issuing '1984':
The book '1984' has been issued successfully.
Returning '1984':
The book '1984' has been returned successfully.
```

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 NUM_GRADES 5
struct Student {
    char name[50];
    int rollNumber;
    float grades[NUM_GRADES];
};
float getHighestGrade(float grades[]) {
    float highest = grades[0];
    for (int i = 1; i < NUM_GRADES; i++) {
        if (grades[i] > highest) {
            highest = grades[i];
        }
    }
    return highest;
}
float getLowestGrade(float grades[]) {
```

```
    float lowest = grades[0];
    for (int i = 1; i < NUM_GRADES; i++) {
        if (grades[i] < lowest) {
            lowest = grades[i];
        }
    }
    return lowest;
}
float getAverageGrade(float  grades[]) {
    float sum = 0;
    for (int i = 0; i < NUM_GRADES; i++) {
        sum += grades[i];
    }
    return sum / NUM_GRADES;
}

int main() {
    struct Student student = {"Sofia", 47, {80.5, 90.0, 85.5, 78.0, 88.5}};

    printf("Student:  %s\n", student.name);
    printf("Roll  Number: %d\n", student.rollNumber);
    printf("Grades:  ");
    for (int i = 0; i < NUM_GRADES; i++) {
        printf("%.2f  ", student.grades[i]);
    }
    printf("\n");
    printf("Highest  Grade: %.2f\n", getHighestGrade(student.grades));
    printf("Lowest  Grade: %.2f\n", getLowestGrade(student.grades));
    printf("Average  Grade: %.2f\n", getAverageGrade(student.grades));
    return 0;
}
```

o/p:

Student: Sofia

Roll Number: 47

Grades: 80.50 90.00 85.50 78.00 88.50

Highest Grade: 90.00

Lowest Grade: 78.00

Average Grade: 84.50

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>
struct Product {
   int productID;
   char name[50];
   int quantity;
   float price;
};

int main() {
   struct Product product = {101, "Laptop", 50, 500.0};

   int soldQuantity;
   float totalSales;
   printf("Product ID: %d\n", product.productID);
   printf("Product Name: %s\n", product.name);
   printf("Available Quantity: %d\n", product.quantity);
   printf("Price per Unit: %.2f\n", product.price);

   printf("Enter quantity sold: ");
```

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

    if (soldQuantity <= product.quantity) {
        product.quantity -= soldQuantity;
        totalSales = soldQuantity * product.price;
        printf("\nUpdated Product Details:\n");
        printf("Remaining Quantity: %d\n", product.quantity);
        printf("Total Sales Value: %.2f\n", totalSales);
    } else {
        printf("Not enough stock available!\n");
    }
    return 0;
}
```

o/p:

```
Product ID: 101
Product Name: Laptop
Available Quantity: 50
Price per Unit: 500.00
Enter quantity sold: 20

Updated Product Details:
Remaining Quantity: 30
Total Sales Value: 10000.00
```

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.

```
#include <stdio.h>
#include <math.h>
struct Point {
```

```
    float x;
    float y;
};
float calculateDistance(struct Point p1, struct Point p2) {
    return sqrt((p2.x - p1.x) * (p2.x - p1.x) + (p2.y - p1.y) * (p2.y - p1.y));
}

int main() {
    struct Point p1, p2;
    printf("Enter the values of point 1 (x y): ");
    scanf("%f %f", &p1.x, &p1.y);

    printf("Enter the values of point 2 (x y): ");
    scanf("%f %f", &p2.x, &p2.y);

    float distance = calculateDistance(p1, p2);
    printf("The distance between the points is: %.2f\n", distance);
    return 0;
}
```

o/p:

```
Enter the values of point 1 (x y): 4 5
Enter the values of point 2 (x y): 6 7
The distance between the points is: 2.83
```

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>
struct Rectangle {
    float length;
    float width;
};
```

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

int main() {
    struct Rectangle rect;
    printf("Enter the length of the rectangle: ");
    scanf("%f", &rect.length);

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

    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;
}
```

o/p:

```
Enter the length of the rectangle: 56
Enter the width of the rectangle: 34
Area of the rectangle: 1904.00
Perimeter of the rectangle: 180.00
```

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>
struct Movie {
    char title[50];
    char director[100];
    int releaseYear;
    float rating;
};
int main(){
    struct Movie movies[3];

    for (int i = 0; i < 3; i++) {
        printf("\nEnter  details for movie %d:\n", i+1);

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

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

        printf("Release  Year: ");
        scanf("%d",  &movies[i].releaseYear);

        printf("Rating:  ");
        scanf("%f",  &movies[i].rating);
    }
    for (int i = 0; i < 3-1; i++) {
        for (int j = 0; j < 3-i-1; j++) {
            if (movies[j].rating < movies[j+1].rating) {
                struct Movie temp = movies[j];
                movies[j] = movies[j+1];
                movies[j+1] = temp;
```

```
            }
        }
    }
    printf("\nMovies sorted by rating:\n");
    for (int i = 0; i < 3; 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: %.1f\n", movies[i].rating);
    }
    return 0;
}
```

o/p:

```
Enter details for movie 1:
Title: nun
Director: sddwf
Release Year: 2012
Rating: 7

Enter details for movie 2:
Title: veeram
Director: ajith
Release Year: 2010
Rating: 6

Enter details for movie 3:
Title: thupaki
Director: vijay
Release Year: 2021
Rating: 9
```

Movies sorted by rating:

Title: thupaki

Director: vijay

Release Year: 2021

Rating: 9.0

Title: nun

Director: sddwf

Release Year: 2012

Rating: 7.0

Title: veeram

Director: ajith

Release Year: 2010

Rating: 6.0

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>

struct Weather {
    int day;
    int month;
    int year;
    float temperature;
    float humidity;
};
void findHighestTemperature(struct Weather weather[], int numDays) {
    int highestTempIndex = 0;
```

```
    for (int i = 1; i < numDays; i++) {
        if (weather[i].temperature > weather[highestTempIndex].temperature) {
            highestTempIndex = i;
        }
    }

    printf("highest temperature is: %d/%d/%d\n",
           weather[highestTempIndex].day,
           weather[highestTempIndex].month,
           weather[highestTempIndex].year);
    printf("Temperature: %.2f°C, Humidity: %.2f%%\n",
           weather[highestTempIndex].temperature,
           weather[highestTempIndex].humidity);
}

int main() {
    struct Weather weatherData[5] = {
        {1, 1, 2023, 25.5, 65.0},
        {2, 1, 2023, 30.0, 60.0},
        {3, 1, 2023, 28.5, 70.0},
        {4, 1, 2023, 32.0, 55.0},
        {5, 1, 2023, 29.0, 60.0}
    };

    findHighestTemperature(weatherData, 5);
    return 0;
}
```

o/p:

highest temperature is: 4/1/2023

Temperature: 32.00°C, Humidity: 55.00%

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>
struct Fraction {
   int numerator;
   int denominator;
};
void printFraction(struct Fraction frac) {
   printf("%d/%d", frac.numerator, frac.denominator);
}

struct Fraction addFractions(struct Fraction frac1, struct Fraction frac2) {
   struct Fraction result;
   result.numerator = frac1.numerator * frac2.denominator + frac2.numerator * frac1.denominator;
   result.denominator = frac1.denominator * frac2.denominator;
   return result;
}
struct Fraction subtractFractions(struct Fraction frac1, struct Fraction frac2) {
   struct Fraction result;
   result.numerator = frac1.numerator * frac2.denominator - frac2.numerator * frac1.denominator;
   result.denominator = frac1.denominator * frac2.denominator;
   return result;
}
struct Fraction multiplyFractions(struct Fraction frac1, struct Fraction frac2) {
   struct Fraction result;
   result.numerator = frac1.numerator * frac2.numerator;
   result.denominator = frac1.denominator * frac2.denominator;
   return result;
}
```

```
struct Fraction divideFractions(struct  Fraction frac1, struct Fraction frac2) {
    struct Fraction result;
    result.numerator  = frac1.numerator  * frac2.denominator;
    result.denominator  = frac1.denominator  * frac2.numerator;
    return result;
}

int main() {
    struct Fraction frac1, frac2, result;

    printf("Enter  the first fraction (numerator  denominator):  ");
    scanf("%d %d", &frac1.numerator,  &frac1.denominator);

    printf("Enter  the second  fraction (numerator  denominator):  ");
    scanf("%d %d", &frac2.numerator,  &frac2.denominator);

    result = addFractions(frac1,  frac2);
    printf("Addition:  ");
    printFraction(result);
    printf("\n");

    result = subtractFractions(frac1,  frac2);
    printf("Subtraction:  ");
    printFraction(result);
    printf("\n");

    result = multiplyFractions(frac1,  frac2);
    printf("Multiplication:  ");
    printFraction(result);
    printf("\n");
```

```
    result = divideFractions(frac1, frac2);
    printf("Division: ");
    printFraction(result);
    printf("\n");

    return 0;
}
```

o/p:

```
Enter the first fraction (numerator denominator): 3 12
Enter the second fraction (numerator denominator): 5 11
Addition: 93/132
Subtraction: -27/132
Multiplication: 15/132
Division: 33/60
```

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>
struct Laptop {
    char brand[50];
    char model[50];
    char processor[50];
    int RAM; // in GB
    float price; // in USD
};

void printLaptop(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 USD\n\n", laptop.price);
}

int main() {
    struct Laptop laptops[5] = {
        {"Dell", "XPS 13", "Intel i7", 16, 1200.50},
        {"HP", "Pavilion", "AMD Ryzen 5", 8, 650.75},
        {"Apple", "MacBook Air", "Apple M1", 8, 999.99},
        {"Lenovo", "ThinkPad", "Intel i5", 16, 950.00},
        {"Acer", "Aspire 5", "Intel i3", 4, 400.00}
    };

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

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

    printf("\nLaptops within the price range %.2f - %.2f USD:\n", minPrice, maxPrice);
    int found = 0;

    for (int i = 0; i < 5; i++) {
        if (laptops[i].price >= minPrice && laptops[i].price <= maxPrice) {
            printLaptop(laptops[i]);
            found = 1;
        }
    }

    if (!found) {
        printf("No laptops found in the specified price range.\n");
```

```
    }
    return 0;
}
```

o/p:

Enter the minimum price: 700

Enter the maximum price: 1300

Laptops within the price range 700.00 - 1300.00 USD:

Brand: Dell

Model: XPS 13

Processor: Intel i7

RAM: 16 GB

Price: 1200.50 USD

Brand: Apple

Model: MacBook Air

Processor: Apple M1

RAM: 8 GB

Price: 999.99 USD

Brand: Lenovo

Model: ThinkPad

Processor: Intel i5

RAM: 16 GB

Price: 950.00 USD

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.

Sol: #include <stdio.h>

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

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

int main() {
   struct Attendance students[] = {
      {101, 50, 45},
      {102, 60, 50},
      {103, 55, 40}
   };
   int n = sizeof(students) / sizeof(students[0]);
 displayAttendance(students, n);
  return 0;
}
```

O/p: Attendance Percentage for Each Student:

Student ID: 101, Attendance: 90.00%

Student ID: 102, Attendance: 83.33%

Student ID: 103, Attendance: 72.73%

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.

Sol:

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

struct Flight {
    char flightNumber[10];
    char departure[30];
    char destination[30];
    double duration;  // in hours
};

void displayShortFlights(struct Flight flights[], int n, double maxDuration) {
    printf("Flights with duration less than %.2f hours:\n", maxDuration);
    int found = 0;
    for (int i = 0; i < n; i++) {
        if (flights[i].duration < maxDuration) {
            printf("Flight: %s, Departure: %s, Destination: %s, Duration: %.2f hours\n",
                flights[i].flightNumber, flights[i].departure, flights[i].destination, flights[i].duration);
            found = 1;
        }
    }
    if (!found) {
        printf("No flights found with duration less than %.2f hours.\n", maxDuration);
    }
}

int main() {
    struct Flight flights[] = {
        {"AA101", "New York", "London", 7.5},
        {"DL202", "Los Angeles", "Tokyo", 11.0},
        {"UA303", "Chicago", "Toronto", 1.5}
```

```
    };
    int n = sizeof(flights) / sizeof(flights[0]);

    double maxDuration;
    printf("Enter maximum flight duration (hours): ");
    scanf("%lf", &maxDuration);

    displayShortFlights(flights, n, maxDuration);

    return 0;
}
```

O/p: Enter maximum flight duration (hours): 7

Flights with duration less than 7.00 hours:

Flight: UA303, Departure: Chicago, Destination: Toronto, Duration: 1.50 hours

Polynomial Representation:

- Define a structure to represent a term of a polynomial (coefficient and exponent).
- Write functions to add and multiply two polynomials.

Sol:

```
#include <stdio.h>

struct Term {
    int coeff, exp;
};

void addPolynomials(struct Term p1[], int n1, struct Term p2[], int n2) {
    int i = 0, j = 0;
    while (i < n1 && j < n2) {
        if (p1[i].exp > p2[j].exp) printf("%dx^%d ", p1[i].coeff, p1[i].exp), i++;
        else if (p1[i].exp < p2[j].exp) printf("%dx^%d ", p2[j].coeff, p2[j].exp), j++;
        else { printf("%dx^%d ", p1[i].coeff + p2[j].coeff, p1[i].exp); i++; j++; }
    }
    while (i < n1) printf("%dx^%d ", p1[i].coeff, p1[i].exp), i++;
```

```
    while (j < n2) printf("%dx^%d ", p2[j].coeff, p2[j].exp), j++;
    printf("\n");
}

void multiplyPolynomials(struct Term p1[], int n1, struct Term p2[], int n2) {
    for (int i = 0; i < n1; i++) {
        for (int j = 0; j < n2; j++) {
            printf("%dx^%d ", p1[i].coeff * p2[j].coeff, p1[i].exp + p2[j].exp);
        }
    }
    printf("\n");
}

int main() {
    struct Term p1[] = {{3, 2}, {5, 1}}, p2[] = {{4, 2}, {1, 1}};
    printf("Sum: ");
    addPolynomials(p1, 2, p2, 2);
    printf("Product: ");
    multiplyPolynomials(p1, 2, p2, 2);
    return 0;
}
```

O/p: Sum: 7x^2 6x^1

Product: 12x^4 3x^3 20x^3 5x^2

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.

Sol:

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

struct Patient {
```

```
    char name[50];
    int age;
    char diagnosis[100];
    char treatment[100];
};

void searchByDiagnosis(struct Patient patients[], int n, const char* diagnosis) {
    printf("Patients with diagnosis '%s':\n", diagnosis);
    int found = 0;
    for (int i = 0; i < n; i++) {
        if (strstr(patients[i].diagnosis, diagnosis)) {
            printf("Name: %s, Age: %d, Treatment: %s\n", patients[i].name, patients[i].age,
patients[i].treatment);
            found = 1;
        }
    }
    if (!found) printf("No patients found with this diagnosis.\n");
}

int main() {
    struct Patient patients[] = {
        {"Alice", 30, "Flu", "Rest and fluids"},
        {"Bob", 45, "Covid", "Antiviral medication"},
        {"Charlie", 60, "Flu", "Antibiotics"},
    };
    int n = sizeof(patients) / sizeof(patients[0]);

    char diagnosis[100];
    printf("Enter diagnosis to search: ");
    scanf("%s", diagnosis);
```

```
    searchByDiagnosis(patients, n, diagnosis);

    return 0;
}
```

O/p: Enter diagnosis to search: Flu

Patients with diagnosis 'Flu':

Name: Alice, Age: 30, Treatment: Rest and fluids

Name: Charlie, Age: 60, Treatment: Antibiotics

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.

Sol:

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

struct Player {
    char name[50];
    char game[50];
    int score;
};

void topScorer(struct Player players[], int n) {
    char games[10][50]; // Store unique games
    int gameCount = 0;

    // Find unique games
    for (int i = 0; i < n; i++) {
        int found = 0;
        for (int j = 0; j < gameCount; j++) {
            if (strcmp(players[i].game, games[j]) == 0) {
                found = 1;
```

```
            break;
        }
    }
    if (!found) {
        strcpy(games[gameCount], players[i].game);
        gameCount++;
    }
}

// Find and display top scorer for each game
for (int i = 0; i < gameCount; i++) {
    int maxScore = -1;
    char topPlayer[50];
    for (int j = 0; j < n; j++) {
        if (strcmp(players[j].game, games[i]) == 0 && players[j].score > maxScore) {
            maxScore = players[j].score;
            strcpy(topPlayer, players[j].name);
        }
    }
    printf("Top scorer for %s: %s with score %d\n", games[i], topPlayer, maxScore);
}
}

int main() {
    struct Player players[] = {
        {"Alice", "Basketball", 25},
        {"Bob", "Basketball", 30},
        {"Charlie", "Football", 40},
        {"David", "Football", 35}
    };
    int n = sizeof(players) / sizeof(players[0]);
```

```
    topScorer(players, n);

    return 0;
}
```

O/p:

Top scorer for Basketball: Bob with score 30

Top scorer for Football: Charlie with score 40

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.

Sol:

```
#include <stdio.h>

struct City {
    char name[50];
    int population;
    float area; // in square kilometers
};

void displayDensity(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/km²\n", cities[i].name, density);
    }
}

int main() {
    struct City cities[] = {
        {"New York", 8419600, 783.8},
        {"Los Angeles", 3980400, 1302},
```

```
    {"Chicago", 2716000, 589}
  };
  int n = sizeof(cities) / sizeof(cities[0]);

  displayDensity(cities, n);

  return 0;
}
```

O/p: City: New York, Population Density: 10742.03 people/km²

City: Los Angeles, Population Density: 3057.14 people/km²

City: Chicago, Population Density: 4611.21 people/km²

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.

Sol:

```
#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) {
  printf("Vehicles registered in %d:\n", year);
  int found = 0;
  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);
```

```
            found = 1;
        }
    }
    if (!found) printf("No vehicles found for this year.\n");
}

int main() {
    struct Vehicle vehicles[] = {
        {"ABC123", "John Doe", "Toyota", 2020},
        {"XYZ789", "Alice Smith", "Honda", 2021},
        {"LMN456", "Bob Johnson", "Ford", 2020}
    };
    int n = sizeof(vehicles) / sizeof(vehicles[0]);

    int year;
    printf("Enter year to search for registered vehicles: ");
    scanf("%d", &year);

    listVehiclesByYear(vehicles, n, year);

    return 0;
}
```

O/p: Enter year to search for registered vehicles: 2020

Vehicles registered in 2020:

Reg No: ABC123, Owner: John Doe, Make: Toyota

Reg No: LMN456, Owner: Bob Johnson, Make: Ford

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.

Sol: #include <stdio.h>

```
#include <string.h>

struct MenuItem {
    char name[50];
    char category[50];
    float price;
};

void displayItemsByCategory(struct MenuItem menu[], int n, const char* category) {
    printf("Menu items in category '%s':\n", category);
    int found = 0;
    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);
            found = 1;
        }
    }
    if (!found) {
        printf("No items found in this category.\n");
    }
}

int main() {
    struct MenuItem menu[] = {
        {"Burger", "Fast Food", 5.99},
        {"Pizza", "Fast Food", 8.99},
        {"Pasta", "Italian", 12.99},
        {"Salad", "Vegetarian", 6.49}
    };
    int n = sizeof(menu) / sizeof(menu[0]);
```

```
    char category[50];
    printf("Enter category to display items: ");
    scanf("%s", category);

    displayItemsByCategory(menu, n, category);

    return 0;
}
```

O/p: Enter category to display items: T Italian

Menu items in category 'Italian':

Name: Pasta, Price: 12.99

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.

Sol:

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

struct SportsTeam {
    char teamName[50];
    char sport[50];
    int numPlayers;
    char coach[50];
};

void displayTeamsBySport(struct SportsTeam teams[], int n, const char* sport) {
    printf("Teams playing sport '%s':\n", sport);
    int found = 0;
    for (int i = 0; i < n; i++) {
        if (strcmp(teams[i].sport, sport) == 0) {
            printf("Team Name: %s, Players: %d, Coach: %s\n", teams[i].teamName, teams[i].numPlayers,
teams[i].coach);
```

```
            found = 1;
        }
    }
    if (!found) {
        printf("No teams found for this sport.\n");
    }
}

int main() {
    struct SportsTeam teams[] = {
        {"Warriors", "Basketball", 12, "Steve Kerr"},
        {"Lions", "Football", 11, "John Doe"},
        {"Spartans", "Basketball", 12, "Tom Smith"},
        {"Eagles", "Football", 11, "Mike Johnson"}
    };
    int n = sizeof(teams) / sizeof(teams[0]);

    char sport[50];
    printf("Enter sport to display teams: ");
    scanf("%s", sport);

    displayTeamsBySport(teams, n, sport);

    return 0;
}
```

O/p:

```
Enter sport to display teams: Football
Teams playing sport 'Football':
Team Name: Lions, Players: 11, Coach: John Doe
Team Name: Eagles, Players: 11, Coach: Mike Johnson
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.

Sol:

```
#include <stdio.h>

struct Student {
    char name[50];
    int marks[5];  // Marks in 5 subjects
    int total;
    float percentage;
};

void calculateMarks(struct Student* student) {
    student->total = 0;
    for (int i = 0; i < 5; i++) {
        student->total += student->marks[i];
    }
    student->percentage = (float)student->total / 5;
}

int main() {
    struct Student students[] = {
        {"Alice", {85, 90, 78, 92, 88}, 0, 0.0},
        {"Bob", {70, 75, 80, 65, 85}, 0, 0.0},
        {"Charlie", {90, 85, 95, 80, 89}, 0, 0.0}
    };

    int n = sizeof(students) / sizeof(students[0]);

    for (int i = 0; i < n; i++) {
        calculateMarks(&students[i]);
```

```
        printf("Student: %s\nTotal Marks: %d\nPercentage: %.2f%%\n\n", students[i].name, students[i].total, students[i].percentage);
    }

    return 0;
}
```

O/p:

Student: Alice

Total Marks: 433

Percentage: 86.60%

Student: Bob

Total Marks: 375

Percentage: 75.00%

Student: Charlie

Total Marks: 439

Percentage: 87.80%

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.

Sol:

```
#include <stdio.h>

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

```
};

void updateStock(struct Product* product, int newStock) {
    product->stock = newStock;
}

float calculateTotalValue(struct Product product) {
    return product.price * product.stock;
}

int main() {
    struct Product products[] = {
        {101, "Laptop", "Electronics", 799.99, 10},
        {102, "Phone", "Electronics", 499.99, 20},
        {103, "Shoes", "Footwear", 59.99, 15}
    };

    int n = sizeof(products) / sizeof(products[0]);

    // Display initial stock and total value
    for (int i = 0; i < n; i++) {
        printf("Product: %s, Stock: %d, Total Value: %.2f\n", products[i].name, products[i].stock,
calculateTotalValue(products[i]));
    }

    // Update stock for product 1 (Laptop)
    updateStock(&products[0], 5); // New stock for Laptop

    // Display updated stock and total value
    printf("\nAfter updating stock:\n");
    for (int i = 0; i < n; i++) {
```

```
    printf("Product: %s, Stock: %d, Total Value: %.2f\n", products[i].name, products[i].stock,
calculateTotalValue(products[i]));
  }

  return 0;
}
```

O/p: Product: Laptop, Stock: 10, Total Value: 7999.90

Product: Phone, Stock: 20, Total Value: 9999.80

Product: Shoes, Stock: 15, Total Value: 899.85

After updating stock:

Product: Laptop, Stock: 5, Total Value: 3999.95

Product: Phone, Stock: 20, Total Value: 9999.80

Product: Shoes, Stock: 15, Total Value: 899.85

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.

Sol:

```
#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, const char* genre) {
  printf("Albums of genre '%s':\n", genre);
  int found = 0;
  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);
            found = 1;
        }
    }
    if (!found) {
        printf("No albums found for this genre.\n");
    }
}

int main() {
    struct Album albums[] = {
        {"Thriller", "Michael Jackson", "Pop", 1982},
        {"Back in Black", "AC/DC", "Rock", 1980},
        {"The Dark Side of the Moon", "Pink Floyd", "Rock", 1973},
        {"Future Nostalgia", "Dua Lipa", "Pop", 2020}
    };

    int n = sizeof(albums) / sizeof(albums[0]);

    char genre[50];
    printf("Enter genre to display albums: ");
    scanf("%s", genre);

    displayAlbumsByGenre(albums, n, genre);

    return 0;
}
```

O/p: Enter genre to display albums: Rock

Albums of genre 'Rock':

Album: Back in Black, Artist: AC/DC, Year: 1980

Album: The Dark Side of the Moon, Artist: Pink Floyd, Year: 1973

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.

Sol:

```
#include <stdio.h>

struct Ticket {
    char movieName[50];
    int seatNumber;
    float price;
};

float totalRevenue = 0;

void bookTicket(struct Ticket* ticket, float price) {
    printf("Enter movie name: ");
    getchar(); // to clear the newline from previous input
    fgets(ticket->movieName, 50, stdin);
    ticket->movieName[strcspn(ticket->movieName, "\n")] = 0; // remove newline character
    printf("Enter seat number: ");
    scanf("%d", &ticket->seatNumber);
    ticket->price = price;
    totalRevenue += ticket->price;
    printf("Ticket booked for Movie: %s, Seat: %d, Price: %.2f\n", ticket->movieName, ticket->seatNumber, ticket->price);
}

int main() {
    struct Ticket tickets[5]; // Assume max 5 tickets for simplicity
    int n = 5;
```

```
    float price = 12.50;  // Price for each ticket

    for (int i = 0; i < n; i++) {
        printf("\nBooking ticket %d\n", i + 1);
        bookTicket(&tickets[i], price);
    }

    printf("\nTotal Revenue Generated: %.2f\n", totalRevenue);

    return 0;
}
```

O/p:

```
Booking ticket 1
Enter movie name: pushpa 2
Enter seat number: 12
Ticket booked for Movie: ushpa 2, Seat: 12, Price: 12.50

Booking ticket 2
Enter movie name: gamechanger
Enter seat number: 25
Ticket booked for Movie: gamechanger, Seat: 25, Price: 12.50

Booking ticket 3
Enter movie name: abc
Enter seat number: 78
Ticket booked for Movie: abc, Seat: 78, Price: 12.50

Booking ticket 4
Enter movie name: xyz
Enter seat number: 56
Ticket booked for Movie: xyz, Seat: 56, Price: 12.50
```

Booking ticket 5

Enter movie name: ram

Enter seat number: 78

Ticket booked for Movie: ram, Seat: 78, Price: 12.50

Total Revenue Generated: 62.50

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.

Sol:

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

// Structure to represent a course
struct Course {
    char courseCode[10];
    char courseName[100];
    char instructor[50];
    int credits;
};

// Function to list all courses taught by a specific instructor
void listCoursesByInstructor(struct Course courses[], int numCourses, char instructor[]) {
    int found = 0;

    printf("Courses taught by %s:\n", instructor);

    // Loop through all courses to find the ones taught by the specified instructor
    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\n", courses[i].credits);
            found = 1; // At least one course found
        }
    }

    // If no courses were found
    if (!found) {
        printf("No courses found for instructor %s.\n", instructor);
    }
}

int main() {
    // Array of course data for 5 courses
    struct Course courses[5] = {
        {"CS101", "Introduction to Programming", "Dr. Smith", 4},
        {"CS102", "Data Structures", "Dr. Smith", 3},
        {"MATH101", "Calculus I", "Dr. Johnson", 4},
        {"CS103", "Algorithms", "Dr. Smith", 3},
        {"PHYS101", "Physics I", "Dr. Williams", 3}
    };

    char instructor[50];

    // Take input for the instructor's name
    printf("Enter the name of the instructor: ");
    fgets(instructor, sizeof(instructor), stdin);
    instructor[strcspn(instructor, "\n")] = 0; // Remove trailing newline character from input

    // List courses taught by the specified instructor
```

```
    listCoursesByInstructor(courses,  5, instructor);


    return 0;
}
```

O/p: Enter the name of the instructor: Dr. Williams

Courses taught by Dr. Williams:

Course Code: PHYS101

Course Name: Physics I

Credits: 3