STRUCTURES -

```
#include <stdio.h>

//structure declaration or structure blueprint
struct Date{
   int date;
   int month;
   int year;

};

struct Time{
   int hour;
   int minutes;
   int sec;

};

struct date_time{
   struct Date currentDate;
   struct Time currentTime;
};

int main()
{
```

```
  struct date_time event;
  event.currentDate.date = 8;
  event.currentTime.hour=12;

  printf("currentDate = %d and currentTime = %d",
event.currentDate.date,event.currentTime.hour);
   return 0;
}

////////----------------------
#include <stdio.h>

struct Date {
   int month;
   int day;
   int year;
};

int main() {

   struct Date TodayDate;
   struct Date *pdate;

   pdate= &TodayDate;
   pdate->month = 1;
   pdate ->day = 8;
   pdate->year = 2025;
```

```
   printf("today month = %d-%d-%d", pdate ->day,pdate->month,pdate->year);

    return 0;
}
```

Nested structures

Both date and time----

```
#include <stdio.h>

//structure declaration or structure blueprint
struct Date{
    int date;
    int month;
    int year;

};

struct Time{
    int hour;
    int minutes;
    int sec;
```

```
};

struct date_time{
   struct Date currentDate;
   struct Time currentTime;
};

int main()
{
  struct date_time event;
  event.currentDate.date = 8;
  event.currentTime.hour=12;

  printf("currentDate = %d and currentTime = %d",
event.currentDate.date,event.currentTime.hour);
   return 0;
}
```

Program to print all moths with respective no of days

```
#include <stdio.h>

struct month {
   int numberofdays;
```

```
    char name[3];
};

int main() {
    struct month months[12] = {
        {31, "JAN"}, {28, "FEB"}, {31, "MAR"}, {30, "APR"},{31, "MAY"}, {30, "JUN"}, {31, "JUL"}, {31, "AUG"},
        {30, "SEP"}, {31, "OCT"}, {30, "NOV"}, {31, "DEC"}
    };

    for (int i = 0; i < 12; i++)
    {
        printf("%s : %d days\n", months[i].name, months[i].numberofdays);
    }

    return 0;
}
```

STRUCTURE AND POINTERS

```
#include <stdio.h>

struct Date {
    int month;
    int day;
    int year;
```

```
};

int main() {

    struct Date TodayDate;
    struct Date *pdate;

    pdate= &TodayDate;
    pdate->month = 1;
    pdate ->day = 8;
    pdate->year = 2025;


   printf("today month = %d-%d-%d", pdate ->day,pdate->month,pdate->year);

    return 0;
}
```

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>
#include <string.h>
#define MAX_STUDENTS 10


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


void addStudent(struct Student students[], int *count);
void displayStudents(struct Student students[], int count);
void findStudent(struct Student students[], int count);
void calculateAverageMarks(struct Student students[], int count);

int main() {
    struct Student students[MAX_STUDENTS];
    int count = 0;
    int choice;

    do {
        printf("\n1. Add Student\n");
        printf("2. Display All Students\n");
```

```
    printf("3. Find Student by Roll Number\n");
    printf("4. Calculate Average Marks\n");
    printf("5. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            addStudent(students, &count);
            break;
        case 2:
            displayStudents(students, count);
            break;
        case 3:
            findStudent(students, count);
            break;
        case 4:
            calculateAverageMarks(students, count);
            break;
        case 5:
            printf("Exiting program.\n");
            break;
        default:
            printf("Invalid choice. Please try again.\n");
    }
} while (choice != 5);
```

```
    return 0;
}


void addStudent(struct Student students[], int *count) {
    if (*count >= MAX_STUDENTS) {
        printf("Maximum student limit reached.\n");
        return;
    }
    printf("Enter name: ");
    scanf(" %[^\n]", students[*count].name);
    printf("Enter roll number: ");
    scanf("%d", &students[*count].rollNumber);
    printf("Enter marks: ");
    scanf("%f", &students[*count].marks);
    (*count)++;
    printf("Student added successfully!\n");
}


void displayStudents(struct Student students[], int count) {
    if (count == 0) {
        printf("No records available.\n");
        return;
    }
```

```
    printf("\n%-20s %-15s %-10s\n", "Name", "Roll Number", "Marks");
    for (int i = 0; i < count; i++) {
        printf("%-20s %-15d %-10.2f\n", students[i].name, students[i].rollNumber,
students[i].marks);
    }
}


void findStudent(struct Student students[], int count) {
    int rollNumber;
    printf("Enter roll number: ");
    scanf("%d", &rollNumber);
    for (int i = 0; i < count; i++) {
        if (students[i].rollNumber == rollNumber) {
            printf("Name: %s, Roll Number: %d, Marks: %.2f\n", students[i].name,
students[i].rollNumber, students[i].marks);
            return;
        }
    }
    printf("Student with roll number %d not found.\n", rollNumber);
}


void calculateAverageMarks(struct Student students[], int count) {
    if (count == 0) {
        printf("No records to calculate average marks.\n");
        return;
```

```
    }
    float totalMarks = 0;
    for (int i = 0; i < count; i++) {
        totalMarks += students[i].marks;
    }
    printf("Average Marks: %.2f\n", totalMarks / count);
}
```

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>

struct Employee{
```

```
    char name[20];
    int id;
    float salary;
    char department[20];
};

void displaySalary(struct Employee employees[],int count,float threshold){

    printf("Employees with salary above %.2f:\n", threshold);
    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\n", employees[i].department);
        }
    }

}

int main(){

    struct Employee employees[]= {
        {"Ab",1000,20000,"embedded"},
         {"Bc",1000,25000,"embedded"},
          {"Cd",1000,27000,"IT"},
```

```
        {"De",1000,26000,"embedded"}

    };

    int count = sizeof(employees)/sizeof(employees[0]);
    float salaryThreshold;

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

     displaySalary(employees, count, salaryThreshold);

     return 0;

}
```

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[100];
    char author[100];
    char ISBN[13];
    float price;
};


void addBook(struct Book inventory[], int *count) {
    printf("Enter the title: ");
    scanf(" %s", inventory[*count].title);
    printf("Enter the author: ");
    scanf(" %s", inventory[*count].author);
    printf("Enter the ISBN: ");
    scanf("%s", inventory[*count].ISBN);
    printf("Enter the price: ");
    scanf("%f", &inventory[*count].price);
    (*count)++;
}


void searchBookByTitle(struct Book inventory[], int count) {
    char searchTitle[100];
    printf("Enter the title to search: ");
    scanf(" %[^\n]", searchTitle);

    printf("Search results for \"%s\":\n", searchTitle);
```

```
    for (int i = 0; i < count; i++) {
        if (strstr(inventory[i].title, searchTitle) != NULL) {
            printf("Title: %s\n", inventory[i].title);
            printf("Author: %s\n", inventory[i].author);
            printf("ISBN: %s\n", inventory[i].ISBN);
            printf("Price: %.2f\n\n", inventory[i].price);
        }
    }
}

int main() {
    struct Book inventory[100];
    int count = 0;
    int choice;

    while (1) {

        printf("\nBook Store Inventory Menu:\n");
        printf("1. Add Book\n");
        printf("2. Search Book by Title\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);


        switch (choice) {
```

```
            case 1:
                addBook(inventory, &count);
                break;
            case 2:
                searchBookByTitle(inventory, count);
                break;
            case 3:
                printf("Exiting the program.\n");
                return 0;
            default:
                printf("Invalid choice! Please try again.\n");
                break;
        }
    }

    return 0;
}
```

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 {
    double real;
    double imaginary;
};

struct Complex addComplex(struct Complex a, struct Complex b) {
    struct Complex result;
    result.real = a.real + b.real;
    result.imaginary = a.imaginary + b.imaginary;
    return result;
}

struct Complex subtractComplex(struct Complex a, struct Complex b) {
    struct Complex result;
    result.real = a.real - b.real;
    result.imaginary = a.imaginary - b.imaginary;
    return result;
}

struct Complex multiplyComplex(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;


    printf("Enter the real part of the first complex number: ");
    scanf("%lf", &num1.real);
    printf("Enter the imaginary part of the first complex number: ");
    scanf("%lf", &num1.imaginary);


    printf("Enter the real part of the second complex number: ");
    scanf("%lf", &num2.real);
    printf("Enter the imaginary part of the second complex number: ");
    scanf("%lf", &num2.imaginary);


    result = addComplex(num1, num2);
    printf("Addition: %.2lf + %.2lfi\n", result.real, result.imaginary);
    result = subtractComplex(num1, num2);
    printf("Subtraction: %.2lf + %.2lfi\n", result.real, result.imaginary);
```

```
    result = multiplyComplex(num1, num2);
    printf("Multiplication: %.2lf + %.2lfi\n", result.real, result.imaginary);

    return 0;
}
```

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>

struct BankAccount {

    int accountNumber;
    char accountHolderName[100];
    double balance;
};

void displayAccountDetails(struct BankAccount account) {
```

```
    printf("Account Number: %d\n", account.accountNumber);
    printf("Account Holder Name: %s\n", account.accountHolderName);
    printf("Balance: %.2lf\n", account.balance);
}


void deposit(struct BankAccount *account, double amount) {

    if (amount > 0) {
        account->balance += amount;
        printf("Deposited %.2lf into the account.\n", amount);
        displayAccountDetails(*account);
    } else {
        printf("Invalid deposit amount!\n");
    }
}


void withdraw(struct BankAccount *account, double amount)
{

    if (amount > 0 && account->balance >= amount) {
        account->balance -= amount;
        printf("Withdrew %.2lf from the account.\n", amount);
        displayAccountDetails(*account);
```

```
    } else {
        printf("Invalid withdrawal amount or insufficient balance!\n");
    }
}

int main() {


    struct BankAccount account;


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


    displayAccountDetails(account);


    double depositAmount;
    printf("Enter amount to deposit: ");
    scanf("%lf", &depositAmount);
    deposit(&account, depositAmount);
```

```
    double withdrawAmount;
    printf("Enter amount to withdraw: ");
    scanf("%lf", &withdrawAmount);
    withdraw(&account, withdrawAmount);

    return 0;
}
```

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>


struct Car {
    char make[50];
    char model[50];
    int year;
    double price;
};
```

```
void printCarsInPriceRange(struct Car cars[], int count, double minPrice, double
maxPrice) {
    printf("Cars within the price range %.2lf to %.2lf:\n", minPrice, maxPrice);
    for (int i = 0; i < count; i++) {
        if (cars[i].price >= minPrice && cars[i].price <= maxPrice) {
            printf("Make: %s\n", cars[i].make);
            printf("Model: %s\n", cars[i].model);
            printf("Year: %d\n", cars[i].year);
            printf("Price: %.2lf\n\n", cars[i].price);
        }
    }
}

int main() {
    struct Car inventory[100];
    int count = 0;
    int numCars;



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



    for (int i = 0; i < numCars; i++) {
        printf("\nEnter details of car %d:\n", i + 1);
        printf("company: ");
```

```
        scanf(" %s", inventory[i].make);
        printf("Model: ");
        scanf(" %s", inventory[i].model);
        printf("Year: ");
        scanf("%d", &inventory[i].year);
        printf("Price: ");
        scanf("%lf", &inventory[i].price);
        count++;
    }


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


    printCarsInPriceRange(inventory, count, minPrice, maxPrice);


    return 0;
}
```

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>



struct Student {
    char name[100];
    int rollNumber;
    int grades[5];
};



void calculateAndDisplayGrades(struct Student student)
{
    int highest = student.grades[0];
    int lowest = student.grades[0];
    int sum = 0;



    for (int i = 0; i < 5; i++) {
        if (student.grades[i] > highest) {
            highest = student.grades[i];
        }
        if (student.grades[i] < lowest) {
            lowest = student.grades[i];
        }
        sum += student.grades[i];
```

```
    }

    double average = sum / 5.0;



    printf("Student Name: %s\n", student.name);
    printf("Roll Number: %d\n", student.rollNumber);
    printf("Highest Grade: %d\n", highest);
    printf("Lowest Grade: %d\n", lowest);
    printf("Average Grade: %.2f\n", average);
}

int main() {
    struct Student student;



    printf("Enter the student's name: ");
    scanf(" %[^\n]", student.name);
    printf("Enter the student's roll number: ");
    scanf("%d", &student.rollNumber);
    printf("Enter the student's grades (5 grades): ");
    for (int i = 0; i < 5; i++) {
        scanf("%d", &student.grades[i]);
    }
```

```
    calculateAndDisplayGrades(student);




    return 0;
}
```

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>
#include <string.h>


struct Product {
    int productID;
    char name[100];
    int quantity;
    double price;
};
```

```
void updateQuantity(struct Product *product, int soldQuantity) {
    if (product->quantity >= soldQuantity) {
        product->quantity -= soldQuantity;
        printf("Sold %d units of %s. Updated quantity: %d\n", soldQuantity, product->name, product->quantity);
    } else {
        printf("Insufficient quantity of %s. Available quantity: %d\n", product->name, product->quantity);
    }
}

double calculateTotalSales(struct Product *product, int soldQuantity) {
    if (product->quantity >= soldQuantity) {
        return soldQuantity * product->price;
    } else {
        printf("Insufficient quantity of %s. Available quantity: %d\n", product->name, product->quantity);
        return 0.0;
    }
}

int main() {
    struct Product product;
    int soldQuantity;
    double totalSalesValue = 0.0;
```

```
printf("Enter product ID: ");
scanf("%d", &product.productID);
printf("Enter product name: ");
scanf(" %[^\n]", product.name);
printf("Enter product quantity: ");
scanf("%d", &product.quantity);
printf("Enter product price: ");
scanf("%lf", &product.price);


printf("\nProduct Details:\n");
printf("Product ID: %d\n", product.productID);
printf("Name: %s\n", product.name);
printf("Quantity: %d\n", product.quantity);
printf("Price: %.2lf\n", product.price);


printf("\nEnter the quantity sold: ");
scanf("%d", &soldQuantity);
updateQuantity(&product, soldQuantity);


totalSalesValue += calculateTotalSales(&product, soldQuantity);
```

```
    printf("Total Sales Value: %.2lf\n", totalSalesValue);

    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.

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

struct Point {
    double x;
    double y;
};
```

```
double 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 point1, point2;
    double distance;


    printf("Enter the coordinates of the first point (x y): ");
    scanf("%lf %lf", &point1.x, &point1.y);


    printf("Enter the coordinates of the second point (x y): ");
    scanf("%lf %lf", &point2.x, &point2.y);


    distance = calculateDistance(point1, point2);

    printf("The distance between the points is: %.2lf\n", distance);

    return 0;
}
```

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>

struct Rectangle {
   double length;
   double width;
};

double calculateArea(struct Rectangle rect) {
   return rect.length * rect.width;
}

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

int main() {
   struct Rectangle rect;

   printf("Enter the length of the rectangle: ");
   scanf("%lf", &rect.length);
```

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



    double area = calculateArea(rect);
    double perimeter = calculatePerimeter(rect);



    printf("Area of the rectangle: %.2lf\n", area);
    printf("Perimeter of the rectangle: %.2lf\n", perimeter);

    return 0;
}
```

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>



struct Movie {
    char title[100];
    char director[100];
    int releaseYear;
```

```
    float rating;
};


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

int main() {
    struct Movie movies[100];
    int count;


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

```
    for (int i = 0; i < count; i++) {
        printf("\nEnter details of movie %d:\n", i + 1);
        printf("Title: ");
        scanf(" %s", movies[i].title);
        printf("Director: ");
        scanf(" %s", movies[i].director);
        printf("Release Year: ");
        scanf("%d", &movies[i].releaseYear);
        printf("Rating: ");
        scanf("%f", &movies[i].rating);
    }


    sortMoviesByRating(movies, count);


    printf("\nMovies sorted by rating:\n");
    for (int i = 0; i < count; i++) {
        printf("Title: %s\n", movies[i].title);
        printf("Director: %s\n", movies[i].director);
        printf("Release Year: %d\n", movies[i].releaseYear);
        printf("Rating: %.2f\n\n", movies[i].rating);
    }

    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>
#include <string.h>

struct WeatherData {
    char date[11];
    double temperature;
    double humidity;
};

void findDayWithHighestTemperature(struct WeatherData data[], int count) {
    int maxTempIndex = 0;

    for (int i = 1; i < count; i++) {
        if (data[i].temperature > data[maxTempIndex].temperature) {
            maxTempIndex = i;
        }
    }
```

```
    printf("The day with the highest temperature is %s with a temperature of
%.2lf°C.\n",
           data[maxTempIndex].date, data[maxTempIndex].temperature);
}

int main() {
    struct WeatherData weather[100];
    int count;



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



    for (int i = 0; i < count; i++) {
        printf("\nEnter data for day %d:\n", i + 1);
        printf("Date (YYYY-MM-DD): ");
        scanf("%s", weather[i].date);
        printf("Temperature (°C): ");
        scanf("%lf", &weather[i].temperature);
        printf("Humidity (%%): ");
        scanf("%lf", &weather[i].humidity);
    }



    findDayWithHighestTemperature(weather, count);
```

```
    return 0;
}
```

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>

// Define the structure for a fraction
struct Fraction {
    int numerator;
    int denominator;
};

// Function to add two fractions
struct Fraction addFractions(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;
    return result;
}

// Function to subtract two fractions
```

```
struct Fraction subtractFractions(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;
    return result;
}

// Function to multiply two fractions
struct Fraction multiplyFractions(struct Fraction f1, struct Fraction f2) {
    struct Fraction result;
    result.numerator = f1.numerator * f2.numerator;
    result.denominator = f1.denominator * f2.denominator;
    return result;
}

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

// Function to simplify a fraction
struct Fraction simplifyFraction(struct Fraction f) {
    int gcd, a = f.numerator, b = f.denominator;
```

```
    // Find greatest common divisor (GCD) using Euclidean algorithm
    while (b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    gcd = a;
    f.numerator /= gcd;
    f.denominator /= gcd;
    return f;
}

// Function to print a fraction
void printFraction(struct Fraction f) {
    f = simplifyFraction(f);
    printf("%d/%d\n", f.numerator, f.denominator);
}

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

    // Get the first fraction from the user
    printf("Enter the numerator and denominator of the first fraction: ");
    scanf("%d %d", &f1.numerator, &f1.denominator);

    // Get the second fraction from the user
```

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

    // Add the fractions
    result = addFractions(f1, f2);
    printf("Addition: ");
    printFraction(result);

    // Subtract the fractions
    result = subtractFractions(f1, f2);
    printf("Subtraction: ");
    printFraction(result);

    // Multiply the fractions
    result = multiplyFractions(f1, f2);
    printf("Multiplication: ");
    printFraction(result);

    // Divide the fractions
    result = divideFractions(f1, f2);
    printf("Division: ");
    printFraction(result);

    return 0;
}
```

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>

// Define the structure to represent a laptop
struct Laptop {
    char brand[50];
    char model[50];
    char processor[50];
    int RAM;
    double price;
};

// Function to list laptops within a specific price range
void listLaptopsInPriceRange(struct Laptop laptops[], int count, double minPrice, double maxPrice) {
    printf("Laptops within the price range %.2lf to %.2lf:\n", minPrice, maxPrice);
    for (int i = 0; i < count; i++) {
        if (laptops[i].price >= minPrice && laptops[i].price <= maxPrice) {
```

```
            printf("\nBrand: %s\n", laptops[i].brand);
            printf("Model: %s\n", laptops[i].model);
            printf("Processor: %s\n", laptops[i].processor);
            printf("RAM: %d GB\n", laptops[i].RAM);
            printf("Price: %.2lf\n", laptops[i].price);
        }
    }
}

int main() {
    struct Laptop inventory[100];
    int count;

    // Get the number of laptops from the user
    printf("Enter the number of laptops: ");
    scanf("%d", &count);

    // Get the details of each laptop from the user
    for (int i = 0; i < count; i++) {
        printf("\nEnter details of laptop %d:\n", i + 1);
        printf("Brand: ");
        scanf(" %[^\n]", inventory[i].brand); // This allows for spaces in the brand name
        printf("Model: ");
        scanf(" %[^\n]", inventory[i].model); // This allows for spaces in the model name
        printf("Processor: ");
```

```
        scanf(" %[^\n]", inventory[i].processor); // This allows for spaces in the processor name
        printf("RAM (in GB): ");
        scanf("%d", &inventory[i].RAM);
        printf("Price: ");
        scanf("%lf", &inventory[i].price);
    }

    // Get the price range from the user
    double minPrice, maxPrice;
    printf("\nEnter the minimum price: ");
    scanf("%lf", &minPrice);
    printf("Enter the maximum price: ");
    scanf("%lf", &maxPrice);

    // List laptops within the specified price range
    listLaptopsInPriceRange(inventory, count, minPrice, maxPrice);

    return 0;
}
```

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

int main() {
    struct Attendance students[100];
    int count;



    printf("Enter the number of students: ");
```

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



    for (int i = 0; i < count; i++) {
        printf("\nEnter details for student %d:\n", i + 1);
        printf("Student ID: ");
        scanf("%d", &students[i].studentID);
        printf("Total Classes: ");
        scanf("%d", &students[i].totalClasses);
        printf("Classes Attended: ");
        scanf("%d", &students[i].classesAttended);
    }



    displayAttendancePercentage(students, count);

    return 0;
}
```

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[50];
    char destination[50];
    int duration;
};


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

int main() {
    struct Flight flights[100];
    int count;
    int maxDuration;
```

```
    printf("Enter the number of flights: ");
    scanf("%d", &count);

    for (int i = 0; i < count; i++) {
        printf("\nEnter details of flight %d:\n", i + 1);
        printf("Flight Number: ");
        scanf(" %[^\n]", flights[i].flightNumber);
        printf("Departure: ");
        scanf(" %[^\n]", flights[i].departure);
        printf("Destination: ");
        scanf(" %[^\n]", flights[i].destination);
        printf("Duration (in minutes): ");
        scanf("%d", &flights[i].duration);
    }

    printf("\nEnter the maximum duration (in minutes): ");
    scanf("%d", &maxDuration);

    displayShortFlights(flights, count, maxDuration);

    return 0;
}
```

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 Term {
   int coefficient;
   int exponent;
};



void addPolynomials(struct Term poly1[], int size1, struct Term poly2[], int size2, struct Term result[], int *resultSize) {


   int i = 0, j = 0, k = 0;
   while (i < size1 && j < size2)
   {
      if (poly1[i].exponent > poly2[j].exponent)
      {
         result[k++] = poly1[i++];
      } else if (poly1[i].exponent < poly2[j].exponent)
      {
```

```
            result[k++] = poly2[j++];
        } else {
            result[k].coefficient = poly1[i].coefficient + poly2[j].coefficient;
            result[k].exponent = poly1[i].exponent;
            k++;
            i++;
            j++;
        }
    }
    while (i < size1) {
        result[k++] = poly1[i++];
    }
    while (j < size2) {
        result[k++] = poly2[j++];
    }
    *resultSize = k;
}



void multiplyPolynomials(struct Term poly1[], int size1, struct Term poly2[], int
size2, struct Term result[], int *resultSize) {

    struct Term temp[100];
    int tempSize = 0;
    *resultSize = 0;
    for (int i = 0; i < size1; i++) {
        for (int j = 0; j < size2; j++) {
```

```
            temp[tempSize].coefficient = poly1[i].coefficient * poly2[j].coefficient;
            temp[tempSize].exponent = poly1[i].exponent + poly2[j].exponent;
            tempSize++;
        }
    }


    for (int i = 0; i < tempSize; i++) {
        int combined = 0;
        for (int j = 0; j < *resultSize; j++) {
            if (result[j].exponent == temp[i].exponent) {
                result[j].coefficient += temp[i].coefficient;
                combined = 1;
                break;
            }
        }
        if (!combined) {
            result[*resultSize] = temp[i];
            (*resultSize)++;
        }
    }
}

int main()
{
```

```
struct Term poly1[] = {{3, 2}, {5, 1}, {6, 0}};
struct Term poly2[] = {{4, 1}, {2, 0}};
struct Term result[100];
int resultSize;

// Add polynomials
addPolynomials(poly1, 3, poly2, 2, result, &resultSize);
printf("Sum: ");
for (int i = 0; i < resultSize; i++) {
    printf("%dx^%d ", result[i].coefficient, result[i].exponent);
    if (i < resultSize - 1) {
        printf("+ ");
    }
}
printf("\n");

// Multiply polynomials
multiplyPolynomials(poly1, 3, poly2, 2, result, &resultSize);
printf("Product: ");
for (int i = 0; i < resultSize; i++) {
    printf("%dx^%d ", result[i].coefficient, result[i].exponent);
    if (i < resultSize - 1) {
        printf("+ ");
    }
}
```

```
    printf("\n");

    return 0;
}
```

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>

// Define the structure to store a patient's medical record
struct MedicalRecord {
    char name[100];
    int age;
    char diagnosis[100];
    char treatment[100];
};

// Function to search for patients by diagnosis
void searchByDiagnosis(struct MedicalRecord records[], int count, const char* diagnosis) {
    printf("Patients with diagnosis '%s':\n", diagnosis);
```

```
    for (int i = 0; i < count; i++) {
        if (strcmp(records[i].diagnosis, diagnosis) == 0) {
            printf("Name: %s\n", records[i].name);
            printf("Age: %d\n", records[i].age);
            printf("Diagnosis: %s\n", records[i].diagnosis);
            printf("Treatment: %s\n\n", records[i].treatment);
        }
    }
}

int main() {
    struct MedicalRecord records[100];
    int count;
    char diagnosis[100];


    printf("Enter the number of patients: ");
    scanf("%d", &count);


    for (int i = 0; i < count; i++) {
        printf("\nEnter details for patient %d:\n", i + 1);
        printf("Name: ");
        scanf(" %[^\n]", records[i].name);
        printf("Age: ");
        scanf("%d", &records[i].age);
```

```
        printf("Diagnosis: ");
        scanf(" %[^\n]", records[i].diagnosis);
        printf("Treatment: ");
        scanf(" %[^\n]", records[i].treatment);
    }


    printf("\nEnter the diagnosis to search for: ");
    scanf(" %[^\n]", diagnosis);


    searchByDiagnosis(records, count, diagnosis);

    return 0;
}
```

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[100];
```

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


void displayTopScorers(struct Player players[], int count) {
    struct Player topScorers[100];
    int topScorersCount = 0;

    for (int i = 0; i < count; i++) {
        int found = 0;
        for (int j = 0; j < topScorersCount; j++) {
            if (strcmp(players[i].game, topScorers[j].game) == 0) {
                found = 1;
                if (players[i].score > topScorers[j].score) {
                    topScorers[j] = players[i];
                }
                break;
            }
        }
        if (!found) {
            topScorers[topScorersCount++] = players[i];
        }
    }

    printf("Top scorers for each game:\n");
```

```
    for (int i = 0; i < topScorersCount; i++) {
        printf("Game: %s\n", topScorers[i].game);
        printf("Name: %s\n", topScorers[i].name);
        printf("Score: %d\n\n", topScorers[i].score);
    }
}

int main() {
    struct Player players[100];
    int count;


    printf("Enter the number of players: ");
    scanf("%d", &count);


    for (int i = 0; i < count; i++) {
        printf("\nEnter details for player %d:\n", i + 1);
        printf("Name: ");
        scanf(" %[^\n]", players[i].name);
        printf("Game: ");
        scanf(" %[^\n]", players[i].game);
        printf("Score: ");
        scanf("%d", &players[i].score);
    }
```

```
    displayTopScorers(players, count);

    return 0;
}
```

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[100];
    int population;
    double area;
};

void displayPopulationDensity(struct City cities[], int count) {
    for (int i = 0; i < count; i++) {
        double density = cities[i].population / cities[i].area;
        printf("City: %s\n", cities[i].name);
        printf("Population: %d\n", cities[i].population);
```

```
        printf("Area: %.2lf sq km\n", cities[i].area);
        printf("Population Density: %.2lf people per sq km\n\n", density);
    }
}

int main() {
    struct City cities[100];
    int count;


    printf("Enter the number of cities: ");
    scanf("%d", &count);


    for (int i = 0; i < count; i++) {
        printf("\nEnter details of city %d:\n", i + 1);
        printf("Name: ");
        scanf(" %[^\n]", cities[i].name);
        printf("Population: ");
        scanf("%d", &cities[i].population);
        printf("Area (in square kilometers): ");
        scanf("%lf", &cities[i].area);
    }


    displayPopulationDensity(cities, count);
```

```
    return 0;
}
```

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 registrationNumber[20];
    char owner[100];
    char make[50];
    int year;
};


void listVehiclesByYear(struct Vehicle vehicles[], int count, int year) {
    printf("Vehicles registered in the year %d:\n", year);
    for (int i = 0; i < count; i++) {
        if (vehicles[i].year == year) {
```

```
            printf("\nRegistration Number: %s\n", vehicles[i].registrationNumber);
            printf("Owner: %s\n", vehicles[i].owner);
            printf("Make: %s\n", vehicles[i].make);
            printf("Year: %d\n", vehicles[i].year);
        }
    }
}

int main() {
    struct Vehicle vehicles[100];
    int count;
    int year;


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


    for (int i = 0; i < count; i++) {
        printf("\nEnter details of vehicle %d:\n", i + 1);
        printf("Registration Number: ");
        scanf(" %[^\n]", vehicles[i].registrationNumber);
        printf("Owner: ");
        scanf(" %[^\n]", vehicles[i].owner);
        printf("Make: ");
        scanf(" %[^\n]", vehicles[i].make);
```

```
        printf("Year: ");
        scanf("%d", &vehicles[i].year);
    }


    printf("\nEnter the year to list vehicles: ");
    scanf("%d", &year);


    listVehiclesByYear(vehicles, count, year);

    return 0;
}
```

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[100];
    char category[50];
    double price;
};


void displayMenuItemsByCategory(struct MenuItem menu[], int count, const
char* category) {
    printf("Menu items in the category '%s':\n", category);
    for (int i = 0; i < count; i++) {
        if (strcmp(menu[i].category, category) == 0) {
            printf("\nName: %s\n", menu[i].name);
            printf("Price: %.2lf\n", menu[i].price);
        }
    }
}

int main() {
    struct MenuItem menu[100];
    int count;
    char category[50];



    printf("Enter the number of menu items: ");
    scanf("%d", &count);
```

```
    for (int i = 0; i < count; i++) {
        printf("\nEnter details for menu item %d:\n", i + 1);
        printf("Name: ");
        scanf(" %[^\n]", menu[i].name);
        printf("Category: ");
        scanf(" %[^\n]", menu[i].category);
        printf("Price: ");
        scanf("%lf", &menu[i].price);
    }


    printf("\nEnter the category to display: ");
    scanf(" %[^\n]", category);


    displayMenuItemsByCategory(menu, count, category);

    return 0;
}
```

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 SportsTeam {
    char teamName[100];
    char sport[50];
    int numberOfPlayers;
    char coach[100];
};


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

```
}

int main() {
    struct SportsTeam teams[100];
    int count;
    char sport[50];


    printf("Enter the number of teams: ");
    scanf("%d", &count);

    for (int i = 0; i < count; i++) {
        printf("\nEnter details for team %d:\n", i + 1);
        printf("Team Name: ");
        scanf(" %s", teams[i].teamName);
        printf("Sport: ");
        scanf(" %s", teams[i].sport);
        printf("Number of Players: ");
        scanf("%d", &teams[i].numberOfPlayers);
        printf("Coach: ");
        scanf(" %s", teams[i].coach);
    }


    printf("\nEnter the sport to display teams: ");
    scanf(" %[^\n]", sport);
```

```
    displayTeamsBySport(teams, count, sport);

    return 0;
}
```

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[100];
    int rollNumber;
    int marks[5];
};

void calculateTotalAndPercentage(struct Student students[], int count) {
    for (int i = 0; i < count; i++) {
        int total = 0;
```

```
        for (int j = 0; j < 5; j++) {
            total += students[i].marks[j];
        }
        double percentage = (total / 5.0);
        printf("Student Name: %s\n", students[i].name);
        printf("Roll Number: %d\n", students[i].rollNumber);
        printf("Total Marks: %d\n", total);
        printf("Percentage: %.2f%%\n\n", percentage);
    }
}

int main() {
    struct Student students[100];
    int count;


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


    for (int i = 0; i < count; i++) {
        printf("\nEnter details for student %d:\n", i + 1);
        printf("Name: ");
        scanf(" %[^\n]", students[i].name);
        printf("Roll Number: ");
        scanf("%d", &students[i].rollNumber);
```

```
        printf("Enter marks for 5 subjects:\n");
        for (int j = 0; j < 5; j++) {
            printf("Subject %d: ", j + 1);
            scanf("%d", &students[i].marks[j]);
        }
    }

    calculateTotalAndPercentage(students, count);

    return 0;
}
```

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[100];
    char category[50];
    double price;
    int stock;
};



void updateStock(struct Product *product, int soldQuantity) {
    if (product->stock >= soldQuantity) {
        product->stock -= soldQuantity;
        printf("Sold %d units of %s. Updated stock: %d\n", soldQuantity, product-
>name, product->stock);
    } else {
        printf("Insufficient stock of %s. Available stock: %d\n", product->name,
product->stock);
    }
}



double calculateTotalStockValue(struct Product products[], int count) {
    double totalValue = 0.0;
    for (int i = 0; i < count; i++) {
```

```
        totalValue += products[i].price * products[i].stock;
    }
    return totalValue;
}

int main() {
    struct Product products[100];
    int count;
    int soldQuantity;

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


    for (int i = 0; i < count; i++) {
        printf("\nEnter details of product %d:\n", i + 1);
        printf("Product ID: ");
        scanf("%d", &products[i].productID);
        printf("Name: ");
        scanf(" %[^\n]", products[i].name);
        printf("Category: ");
        scanf(" %[^\n]", products[i].category);
        printf("Price: ");
        scanf("%lf", &products[i].price);
        printf("Stock: ");
        scanf("%d", &products[i].stock);
```

```
    }


    printf("\nEnter the product ID to update stock: ");
    int productID;
    scanf("%d", &productID);
    printf("Enter the quantity sold: ");
    scanf("%d", &soldQuantity);

    for (int i = 0; i < count; i++) {
        if (products[i].productID == productID) {
            updateStock(&products[i], soldQuantity);
            break;
        }
    }


    double totalStockValue = calculateTotalStockValue(products, count);
    printf("Total value of products in stock: %.2lf\n", totalStockValue);

    return 0;
}
```

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 MusicAlbum {
    char albumName[100];
    char artist[100];
    char genre[50];
    int releaseYear;
};



void displayAlbumsByGenre(struct MusicAlbum albums[], int count, const
char* genre) {
    printf("Albums in the genre '%s':\n", genre);
    for (int i = 0; i < count; i++) {
        if (strcmp(albums[i].genre, genre) == 0) {
            printf("\nAlbum Name: %s\n", albums[i].albumName);
            printf("Artist: %s\n", albums[i].artist);
            printf("Release Year: %d\n", albums[i].releaseYear);
        }
    }
}

int main() {
```

```
struct MusicAlbum albums[100];
int count;
char genre[50];


printf("Enter the number of albums: ");
scanf("%d", &count);


for (int i = 0; i < count; i++) {
    printf("\nEnter details for album %d:\n", i + 1);
    printf("Album Name: ");
    scanf(" %[^\n]", albums[i].albumName);
    printf("Artist: ");
    scanf(" %[^\n]", albums[i].artist);
    printf("Genre: ");
    scanf(" %[^\n]", albums[i].genre);
    printf("Release Year: ");
    scanf("%d", &albums[i].releaseYear);
}


printf("\nEnter the genre to display albums: ");
scanf(" %[^\n]", genre);
```

```
    displayAlbumsByGenre(albums, count, genre);

    return 0;
}
```

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[100];
    int seatNumber;
    double price;
};

void bookTicket(struct Ticket tickets[], int *count, const char* movieName, int seatNumber, double price) {
    strcpy(tickets[*count].movieName, movieName);
    tickets[*count].seatNumber = seatNumber;
    tickets[*count].price = price;
    (*count)++;
```

```
    printf("Ticket booked: Movie: %s, Seat Number: %d, Price: %.2lf\n",
movieName, seatNumber, price);
}



double calculateTotalRevenue(struct Ticket tickets[], int count) {
    double totalRevenue = 0.0;
    for (int i = 0; i < count; i++) {
        totalRevenue += tickets[i].price;
    }
    return totalRevenue;
}

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

    while (1) {

        printf("\nCinema Ticket Booking Menu:\n");
        printf("1. Book a Ticket\n");
        printf("2. Display Total Revenue\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
```

```
    if (choice == 1) {
        char movieName[100];
        int seatNumber;
        double price;

        printf("Enter movie name: ");
        scanf(" %[^\n]", movieName);
        printf("Enter seat number: ");
        scanf("%d", &seatNumber);
        printf("Enter ticket price: ");
        scanf("%lf", &price);


        bookTicket(tickets, &count, movieName, seatNumber, price);
    } else if (choice == 2) {

        double totalRevenue = calculateTotalRevenue(tickets, count);
        printf("Total Revenue: %.2lf\n", totalRevenue);
    } else if (choice == 3) {

        printf("Exiting the program.\n");
        break;
    } else {
        printf("Invalid choice! Please try again.\n");
    }
}
```

```
    return 0;
}
```

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[20];
    char name[100];
    char instructor[100];
    int credits;
};

void listCoursesByInstructor(struct Course courses[], int count, const char* instructor) {
    printf("Courses taught by %s:\n", instructor);
    for (int i = 0; i < count; i++) {
        if (strcmp(courses[i].instructor, instructor) == 0) {
            printf("\nCourse Code: %s\n", courses[i].courseCode);
            printf("Name: %s\n", courses[i].name);
```

```
            printf("Credits: %d\n", courses[i].credits);
        }
    }
}

int main() {
    struct Course courses[100];
    int count;
    char instructor[100];


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


    for (int i = 0; i < count; i++) {
        printf("\nEnter details for course %d:\n", i + 1);
        printf("Course Code: ");
        scanf(" %[^\n]", courses[i].courseCode);
        printf("Name: ");
        scanf(" %[^\n]", courses[i].name);
        printf("Instructor: ");
        scanf(" %[^\n]", courses[i].instructor);
        printf("Credits: ");
        scanf("%d", &courses[i].credits);
    }
```

```
    printf("\nEnter the instructor name to display courses: ");
    scanf(" %[^\n]", instructor);


    listCoursesByInstructor(courses, count, instructor);

    return 0;
}
```