STRUCTURE
==========

```
#include <stdio.h>
//structure declaration//structure blueprint
struct Date{
    int day;
    int months;
    int years;
};

int main()
{
    struct Date todayDate;
    todayDate.day=8;
    todayDate.months=1;
    todayDate.years=2025;
    printf("TodayDate = %d-%d-%d \n",todayDate.day,todayDate.months,todayDate.years);
    struct Date YesterdayDate;
    YesterdayDate.day=7;
    YesterdayDate.months=1;
    YesterdayDate.years=2025;
    printf("YesterdayDate = %d-%d-%d
\n",YesterdayDate.day,YesterdayDate.months,YesterdayDate.years);
    printf("Size of struct date = %ld\n",sizeof(todayDate));
    printf("Address of todayDate =%p\n",&todayDate);
    printf("Size of struct date = %ld\n",sizeof(YesterdayDate));
    printf("Address of YesterdayDate =%p\n",&YesterdayDate);

    return 0;
}

/*****************************************************************************


#include <stdio.h>
//structure declaration//structure blueprint
struct Date{
    int day;
    int months;
    int years;
}todayDate,YesterdayDate;

int main()
{
    //struct Date todayDate;
    todayDate.day=8;
    todayDate.months=1;
    todayDate.years=2025;
    printf("TodayDate = %d-%d-%d \n",todayDate.day,todayDate.months,todayDate.years);
    //struct Date YesterdayDate;
    YesterdayDate.day=7;
    YesterdayDate.months=1;
    YesterdayDate.years=2025;
```

```
    printf("YesterdayDate = %d-%d-%d
\n",YesterdayDate.day,YesterdayDate.months,YesterdayDate.years);
    printf("Size of struct date = %ld\n",sizeof(todayDate));
    printf("Address of todayDate =%p\n",&todayDate);
    printf("Size of struct date = %ld\n",sizeof(YesterdayDate));
    printf("Address of YesterdayDate =%p\n",&YesterdayDate);

    return 0;
}

================================================================================
=====

/*******************************************************************************

Welcome to GDB Online.
  GDB online is an online compiler and debugger tool for C, C++, Python, PHP, Ruby,
  C#, OCaml, VB, Perl, Swift, Prolog, Javascript, Pascal, COBOL, HTML, CSS, JS
  Code, Compile, Run and Debug online from anywhere in world.

*******************************************************************************/
#include <stdio.h>
//structure declaration//structure blueprint
struct Date{
    int day;
    int months;
    int years;
}todayDate,YesterdayDate;

int main()
{
    struct Date todayDate={8,1,2025};

    printf("TodayDate = %d-%d-%d \n",todayDate.day,todayDate.months,todayDate.years);
    //struct Date YesterdayDate;
    YesterdayDate.day=7;
    YesterdayDate.months=1;
    YesterdayDate.years=2025;
    printf("YesterdayDate = %d-%d-%d
\n",YesterdayDate.day,YesterdayDate.months,YesterdayDate.years);
    printf("Size of struct date = %ld\n",sizeof(todayDate));
    printf("Address of todayDate =%p\n",&todayDate);
    printf("Size of struct date = %ld\n",sizeof(YesterdayDate));
    printf("Address of YesterdayDate =%p\n",&YesterdayDate);

    return 0;
}
================================================================================
==
#include <stdio.h>
#include <string.h>

#define MAX_STUDENTS 100

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);
}

================================================================================

#include <stdio.h>
//structure declaration//structure blueprint
struct Date{
    int day;
    int months;
    int years;
}todayDate,YesterdayDate;
```

```
int main()
{
    struct Date myDates[10];
    myDates[3].months=1;
    myDates[3].day=8;
    myDates[3].years=2025;

    printf("TodayDate = %d-%d-%d \n", myDates[3].day,myDates[3].months,myDates[3].years);


    return 0;
}
```

STRUCTURE ARRAY PROGRAM

==========

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

// Define a structure to store the month and number of days
struct Month {
    char name[20];
    int numberOfDays;
};

int main() {
    // Declare an array of structures for 12 months
    struct Month months[12] = {
        {"JAN", 31},
        {"FEB", 28},
        {"MAR", 31},
        {"APR", 30},
        {"MAY", 31},
        {"JUN", 30},
        {"JUL", 31},
        {"AUG", 31},
        {"SEP", 30},
        {"OCT", 31},
        {"NOV", 30},
        {"DEC", 31}
    };

    // Print the name of each month along with the number of days
    printf("Month Name and the Number of Days:\n");
    for (int i = 0; i < 12; i++) {
        printf("%s: %d days\n", months[i].name, months[i].numberOfDays);
    }

    return 0;
}
```

==========

```
#include <stdio.h>

struct Date {
```

```
    int month;
    int day;
    int year;
};

int main() {
    struct Date today, *dateptr;
    dateptr = &today;

    dateptr->month = 9;
    dateptr->day = 14;
    dateptr->year = 2025;

    printf("The day, month, and year are: %d-%d-%d\n", dateptr->day, dateptr->month,
dateptr->year%100);

    return 0;
}
```

STRUCTURE AND FUNCTION

==========

```
#include <stdio.h>

struct num{
    int a,b;
};
int sum(struct num,struct num);
int main(){

  struct num num1,num2;
  num1.a=30;
  num2.a=40;

  int sumA= sum(num1,num2);
   printf("sumA = %d",sumA);
    return 0;
}
int sum(struct num num1,struct num num2){
    int sum = num1.a+num2.a;
    return sum;
}
```

SET OF PROBLEMS

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>

// Define a structure for Employee details
struct Employee {
    char name[50];
```

```
    int id;
    float salary;
    char department[50];
};

// Function to display details of employees whose salary is above a certain threshold
void displayHighSalaryEmployees(struct Employee employees[], int count, float threshold) {
    printf("Employees with salary above %.2f:\n", threshold);
    printf("-----------------------------------------------\n");
    printf("Name\t\tID\tSalary\t\tDepartment\n");
    printf("-----------------------------------------------\n");

    // Iterate through the employee array and display details if salary > threshold
    for (int i = 0; i < count; i++) {
        if (employees[i].salary > threshold) {
            printf("%s\t\t%d\t%.2f\t\t%s\n", employees[i].name, employees[i].id, employees[i].salary,
employees[i].department);
        }
    }
}

int main() {
    // Array of employees
    struct Employee employees[5] = {
        {"Alice", 101, 55000.0, "HR"},
        {"Bob", 102, 75000.0, "Engineering"},
        {"Charlie", 103, 48000.0, "Sales"},
        {"David", 104, 82000.0, "Marketing"},
        {"Eve", 105, 60000.0, "Engineering"}
    };

    // Threshold salary
    float threshold = 60000.0;

    // Call function to display employees with salary above threshold
    displayHighSalaryEmployees(employees, 5, threshold);

    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>

#define MAX_BOOKS 100

// Structure to represent a book
struct Book {
    char title[100];
    char author[100];
    char isbn[20];
    float price;
```

```
};

// Function to add a book to the inventory
void addBook(struct Book inventory[], int *count) {
    if (*count >= MAX_BOOKS) {
        printf("Inventory is full! Cannot add more books.\n");
        return;
    }

    // Using scanf to read strings without fgets
    printf("Enter book title: ");
    scanf("%s", inventory[*count].title); // Note: This will only read one word (no spaces)

    printf("Enter author name: ");
    scanf("%s", inventory[*count].author); // Note: This will only read one word (no spaces)

    printf("Enter ISBN: ");
    scanf("%s", inventory[*count].isbn); // Note: This will only read one word (no spaces)

    printf("Enter price: ");
    scanf("%f", &inventory[*count].price);

    (*count)++;
    printf("Book added successfully!\n");
}

// Function to search a book by title
void searchByTitle(struct Book inventory[], int count, char searchTitle[]) {
    int found = 0;

    for (int i = 0; i < count; i++) {
        if (strstr(inventory[i].title, searchTitle) != NULL) { // Search by title
            printf("\nBook Found:\n");
            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", inventory[i].price);
            found = 1;
        }
    }

    if (!found) {
        printf("No books found with the title \"%s\".\n", searchTitle);
    }
}

// Function to display all books
void displayBooks(struct Book inventory[], int count) {
    if (count == 0) {
        printf("No books in the inventory.\n");
        return;
    }

    printf("\nBook Inventory:\n");
    for (int i = 0; i < count; i++) {
```

```
        printf("\nBook %d:\n", i + 1);
        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", inventory[i].price);
    }
}

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

    while (1) {
        printf("\nBook Store Inventory Management System\n");
        printf("1. Add Book\n");
        printf("2. Search Book by Title\n");
        printf("3. Display All Books\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                addBook(inventory, &count);
                break;
            case 2: {
                char searchTitle[100];
                printf("Enter the title to search: ");
                scanf("%s", searchTitle); // Note: This will only read one word (no spaces)
                searchByTitle(inventory, count, searchTitle);
                break;
            }
            case 3:
                displayBooks(inventory, count);
                break;
            case 4:
                printf("Exiting program...\n");
                return 0;
            default:
                printf("Invalid choice! Please try again.\n");
        }
    }

    return 0;
}
```

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>
#include <stdbool.h>

// Structure to represent a date
```

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

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

// Function to validate the date
bool isValidDate(struct Date date) {
    // Check if month is between 1 and 12
    if (date.month < 1 || date.month > 12) {
        return false;
    }

    // Number of days in each month for non-leap years
    int daysInMonth[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

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

    // Check if the day is valid for the given month
    if (date.day < 1 || date.day > daysInMonth[date.month - 1]) {
        return false;
    }

    return true;
}

int main() {
    struct Date date;

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

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

```
    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>

// Define a structure to represent a complex number
struct Complex {
    float real;  // Real part of the complex number
    float imag;  // Imaginary part of the complex number
};

// Function to add two complex numbers
struct Complex addComplex(struct Complex c1, struct Complex c2) {
    struct Complex result;
    result.real = c1.real + c2.real;  // Add real parts
    result.imag = c1.imag + c2.imag;  // Add imaginary parts
    return result;
}

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

// Function to multiply two complex numbers
struct Complex multiplyComplex(struct Complex c1, struct Complex c2) {
    struct Complex result;
    result.real = (c1.real * c2.real) - (c1.imag * c2.imag);  // Real part of the product
    result.imag = (c1.real * c2.imag) + (c1.imag * c2.real);  // Imaginary part of the product
    return result;
}

// Function to display a complex number
void displayComplex(struct Complex c) {
    if (c.imag >= 0)
        printf("%.2f + %.2fi\n", c.real, c.imag);  // Display as a + bi
    else
        printf("%.2f - %.2fi\n", c.real, -c.imag);  // Display as a - bi
}

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

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

    // Input for the second complex number
```

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

    // Add the two complex numbers
    result = addComplex(c1, c2);
    printf("Sum: ");
    displayComplex(result);

    // Subtract the two complex numbers
    result = subtractComplex(c1, c2);
    printf("Difference: ");
    displayComplex(result);

    // Multiply the two complex numbers
    result = multiplyComplex(c1, c2);
    printf("Product: ");
    displayComplex(result);

    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>

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

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

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

```
// Function to display account information
void displayAccountInfo(struct BankAccount account) {
    printf("\nAccount Number: %d\n", account.accountNumber);
    printf("Account Holder: %s\n", account.accountHolderName);
    printf("Current Balance: $%.2f\n", account.balance);
}

int main() {
    // Create a bank account
    struct BankAccount account1;

    // Assign values to the account
    account1.accountNumber = 123456;
    strcpy(account1.accountHolderName, "John Doe");
    account1.balance = 500.00; // Initial balance

    // Display initial account information
    displayAccountInfo(account1);

    // Deposit some money
    deposit(&account1, 200.00);

    // Withdraw some money
    withdraw(&account1, 100.00);

    // Display updated account information
    displayAccountInfo(account1);

    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>

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

// Function to print car details
void printCar(struct Car car) {
    printf("Make: %s\n", car.make);
    printf("Model: %s\n", car.model);
    printf("Year: %d\n", car.year);
    printf("Price: %.2f\n", car.price);
    printf("----------------------------\n");
}
```

```
int main() {
    int n, i;
    float lowerLimit, upperLimit;

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

    // Declare an array of Car structures
    struct Car cars[n];

    // Input details of each car
    for(i = 0; i < n; i++) {
        printf("\nEnter details for car %d:\n", i + 1);
        printf("Make: ");
        scanf(" %[^\n]%*c", cars[i].make); // to allow spaces in the make
        printf("Model: ");
        scanf(" %[^\n]%*c", cars[i].model); // to allow spaces in the model
        printf("Year: ");
        scanf("%d", &cars[i].year);
        printf("Price: ");
        scanf("%f", &cars[i].price);
    }

    // Input price range
    printf("\nEnter price range:\n");
    printf("Lower limit: ");
    scanf("%f", &lowerLimit);
    printf("Upper limit: ");
    scanf("%f", &upperLimit);

    // Print cars within the specified price range
    printf("\nCars within the price range %.2f to %.2f:\n", lowerLimit, upperLimit);
    for(i = 0; i < n; i++) {
        if(cars[i].price >= lowerLimit && cars[i].price <= upperLimit) {
            printCar(cars[i]);
        }
    }

    return 0;
}
```

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>

// Structure to represent a library book
struct Book {
    char title[100];
    char author[100];
```

```
    int publicationYear;
    char status[10];  // "issued" or "available"
};

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

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

int main() {
    // Create a sample book
    struct Book book1 = {"The C Programming Language", "Brian Kernighan & Dennis Ritchie", 1978,
"available"};

    printf("Book Details:\n");
    printf("Title: %s\n", book1.title);
    printf("Author: %s\n", book1.author);
    printf("Publication Year: %d\n", book1.publicationYear);
    printf("Status: %s\n\n", book1.status);

    // Issue the book
    issueBook(&book1);  // Issuing the book

    // Try to issue again
    issueBook(&book1);  // Trying to issue it again

    // Return the book
    returnBook(&book1);  // Returning the book

    // Try to return again
    returnBook(&book1);  // Trying to return it again

    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>
```

```c
#define MAX_GRADES 5  // Number of grades per student

// Define the structure to store student information
struct Student {
    char name[50];
    int roll_number;
    int grades[MAX_GRADES];
};

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

// Function to find the highest grade
int find_highest_grade(int grades[], int num_grades) {
    int highest = grades[0];
    for (int i = 1; i < num_grades; i++) {
        if (grades[i] > highest) {
            highest = grades[i];
        }
    }
    return highest;
}

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

int main() {
    // Declare an array of students
    struct Student students[3];  // You can adjust the size based on the number of students

    // Input student data
    for (int i = 0; i < 3; i++) {
        printf("Enter details for student %d:\n", i + 1);
        printf("Name: ");
        scanf("%s", students[i].name);  // Read name (no spaces allowed)

        printf("Roll Number: ");
        scanf("%d", &students[i].roll_number);

        printf("Enter grades (5 grades): ");
```

```
        for (int j = 0; j < MAX_GRADES; j++) {
            scanf("%d", &students[i].grades[j]);
        }
    }

    // Display the details and statistics for each student
    for (int i = 0; i < 3; i++) {
        printf("\nStudent Name: %s\n", students[i].name);
        printf("Roll Number: %d\n", students[i].roll_number);

        int highest = find_highest_grade(students[i].grades, MAX_GRADES);
        int lowest = find_lowest_grade(students[i].grades, MAX_GRADES);
        float average = calculate_average(students[i].grades, MAX_GRADES);

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

    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>

// Define the structure for Product
struct Product {
    int productID;
    char name[50];
    int quantity;
    float price;
};

// Function to update quantity after a sale
void updateQuantity(struct Product *product, int soldQuantity) {
    if (soldQuantity > product->quantity) {
        printf("Error: Not enough stock to complete the sale.\n");
    } else {
        product->quantity -= soldQuantity;
        printf("Sale completed. Updated quantity of '%s': %d\n", product->name, product->quantity);
    }
}

// Function to calculate total sales value
float calculateTotalSales(struct Product product, int soldQuantity) {
    return soldQuantity * product.price;
}

int main() {
    // Example product initialization
    struct Product product1 = {101, "Laptop", 50, 999.99};
```

```
    struct Product product2 = {102, "Smartphone", 100, 499.99};

    printf("Product 1: %s, ID: %d, Price: %.2f, Quantity: %d\n",
           product1.name, product1.productID, product1.price, product1.quantity);
    printf("Product 2: %s, ID: %d, Price: %.2f, Quantity: %d\n",
           product2.name, product2.productID, product2.price, product2.quantity);

    // Example sale
    int soldQuantity1 = 10;
    float salesValue1 = calculateTotalSales(product1, soldQuantity1);
    printf("Total sales value for %d '%s': $%.2f\n", soldQuantity1, product1.name, salesValue1);

    // Update the quantity of the product
    updateQuantity(&product1, soldQuantity1);

    // Example sale for another product
    int soldQuantity2 = 5;
    float salesValue2 = calculateTotalSales(product2, soldQuantity2);
    printf("Total sales value for %d '%s': $%.2f\n", soldQuantity2, product2.name, salesValue2);

    // Update the quantity of the second product
    updateQuantity(&product2, soldQuantity2);

    return 0;
}
```

SECOND SET OF PROBLEMS
==========

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>

// Define a structure for a point in 2D space
struct Point {
    float x;
    float y;
};

// Function to calculate the distance between two points
float calculateDistance(struct Point p1, struct Point p2) {
    float distance;
    distance = sqrt(pow(p2.x - p1.x, 2) + pow(p2.y - p1.y, 2));
    return distance;
}

int main() {
    // Declare two points
    struct Point point1, point2;

    // Input coordinates for point1
```

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

    // Input coordinates for point2
    printf("Enter the coordinates of point2 (x2 y2): ");
    scanf("%f %f", &point2.x, &point2.y);

    // Calculate the distance between the two points
    float dist = calculateDistance(point1, point2);

    // Output the result
    printf("The distance between point1 and point2 is: %.2f\n", dist);

    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>

// Define a structure for Rectangle
struct Rectangle {
    float length;
    float width;
};

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

// Function to calculate the perimeter of the rectangle
float calculatePerimeter(struct Rectangle rect) {
    return 2 * (rect.length + rect.width);
}

int main() {
    struct Rectangle rect;

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

    // Calculate and display the area and perimeter
    printf("Area of the rectangle: %.2f\n", calculateArea(rect));
    printf("Perimeter of the rectangle: %.2f\n", calculatePerimeter(rect));

    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>

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

// Function to sort movies by rating in descending order
void sortMovies(struct Movie movies[], int n) {
    struct Movie temp;
    for (int i = 0; i < n-1; i++) {
        for (int j = i+1; j < n; j++) {
            if (movies[i].rating < movies[j].rating) {
                // Swap the movies if the first one has a lower rating
                temp = movies[i];
                movies[i] = movies[j];
                movies[j] = temp;
            }
        }
    }
}

// Function to print the movie details
void printMovies(struct Movie movies[], int n) {
    printf("\nMovie Details (Sorted by Rating):\n");
    for (int i = 0; i < n; 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: %.1f\n\n", movies[i].rating);
    }
}

int main() {
    int n;

    // Ask the user for the number of movies
    printf("Enter the number of movies: ");
    scanf("%d", &n);

    struct Movie movies[n]; // Array of movies

    // Get the movie details from the user
    for (int i = 0; i < n; i++) {
        printf("\nEnter details for movie %d:\n", i+1);
        getchar();  // To clear the buffer for string inputs
        printf("Enter movie title: ");
        fgets(movies[i].title, 100, stdin);
```

```
        movies[i].title[strcspn(movies[i].title, "\n")] = '\0';  // Remove newline character

        printf("Enter director name: ");
        fgets(movies[i].director, 100, stdin);
        movies[i].director[strcspn(movies[i].director, "\n")] = '\0';  // Remove newline character

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

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

    // Sort the movies by rating
    sortMovies(movies, n);

    // Print the sorted movie details
    printMovies(movies, n);

    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>

// Define a structure to store weather data for a day
struct Weather {
    char date[11];  // Date in format YYYY-MM-DD
    float temperature;  // Temperature in Celsius
    int humidity;  // Humidity percentage
};

// Function to find the day with the highest temperature
struct Weather find_highest_temperature(struct Weather weather_data[], int n) {
    struct Weather highest_temp_day = weather_data[0];  // Start with the first day
    for (int i = 1; i < n; i++) {
        if (weather_data[i].temperature > highest_temp_day.temperature) {
            highest_temp_day = weather_data[i];  // Update with the new highest temperature
        }
    }
    return highest_temp_day;
}

int main() {
    // Define an array of weather data for 7 days
    struct Weather weather_data[] = {
        {"2025-01-01", 15.0, 80},
        {"2025-01-02", 18.0, 75},
        {"2025-01-03", 20.0, 70},
        {"2025-01-04", 22.0, 65},
        {"2025-01-05", 17.0, 85},
```

```
        {"2025-01-06", 21.0, 78},
        {"2025-01-07", 19.0, 72}
    };

    // Find the day with the highest temperature
    int n = sizeof(weather_data) / sizeof(weather_data[0]);  // Number of days
    struct Weather highest_temp_day = find_highest_temperature(weather_data, n);

    // Print the result
    printf("The day with the highest temperature is %s with %.1f°C.\n",
           highest_temp_day.date, highest_temp_day.temperature);

    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>

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

// Function to compute the greatest common divisor (GCD)
int gcd(int a, int b) {
    while (b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}

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

// Function to add two fractions
struct Fraction add(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;
    simplify(&result);
    return result;
}

// Function to subtract two fractions
struct Fraction subtract(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;
    simplify(&result);
    return result;
}

// Function to multiply two fractions
struct Fraction multiply(struct Fraction frac1, struct Fraction frac2) {
    struct Fraction result;
    result.numerator = frac1.numerator * frac2.numerator;
    result.denominator = frac1.denominator * frac2.denominator;
    simplify(&result);
    return result;
}

// Function to divide two fractions
struct Fraction divide(struct Fraction frac1, struct Fraction frac2) {
    struct Fraction result;
    if (frac2.numerator == 0) {
        printf("Error! Division by zero.\n");
        result.numerator = 0;
        result.denominator = 1;
        return result;  // return an invalid fraction if division by zero occurs
    }
    result.numerator = frac1.numerator * frac2.denominator;
    result.denominator = frac1.denominator * frac2.numerator;
    simplify(&result);
    return result;
}

// Function to print a fraction
void printFraction(struct Fraction frac) {
    printf("%d/%d", frac.numerator, frac.denominator);
}

int main() {
    struct Fraction frac1 = {3, 4}; // 3/4
    struct Fraction frac2 = {5, 6}; // 5/6

    struct Fraction result;

    // Addition
    result = add(frac1, frac2);
    printf("Sum: ");
    printFraction(result);
    printf("\n");

    // Subtraction
    result = subtract(frac1, frac2);
    printf("Difference: ");
    printFraction(result);
    printf("\n");

    // Multiplication
```

```
    result = multiply(frac1, frac2);
    printf("Product: ");
    printFraction(result);
    printf("\n");

    // Division
    result = divide(frac1, frac2);
    printf("Quotient: ");
    printFraction(result);
    printf("\n");

    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 a structure to represent a laptop
struct Laptop {
    char brand[50];
    char model[50];
    char processor[50];
    int ram; // in GB
    float price; // in USD
};

// Function to list laptops within a specific price range
void listLaptopsInPriceRange(struct Laptop laptops[], int count, float minPrice, float maxPrice) {
    printf("Laptops in the price range %.2f - %.2f:\n", minPrice, maxPrice);
    printf("-----------------------------------------------------\n");

    int found = 0; // To check if any laptop is found in the range

    for (int i = 0; i < count; i++) {
        if (laptops[i].price >= minPrice && laptops[i].price <= maxPrice) {
            printf("Brand: %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: %.2f USD\n", laptops[i].price);
            printf("-----------------------------------------------------\n");
            found = 1;
        }
    }

    if (!found) {
        printf("No laptops found in this price range.\n");
    }
}

int main() {
```

```
    // Create an array of laptops
    struct Laptop laptops[5] = {
        {"Dell", "Inspiron 15", "Intel i5", 8, 650.50},
        {"HP", "Pavilion 14", "AMD Ryzen 5", 16, 800.99},
        {"Apple", "MacBook Air", "Apple M1", 8, 1200.00},
        {"Asus", "ZenBook 13", "Intel i7", 16, 1300.75},
        {"Lenovo", "ThinkPad X1", "Intel i7", 32, 2000.00}
    };

    // Define price range
    float minPrice, maxPrice;

    // Get price range from user
    printf("Enter minimum price: ");
    scanf("%f", &minPrice);
    printf("Enter maximum price: ");
    scanf("%f", &maxPrice);

    // Call the function to list laptops in the price range
    listLaptopsInPriceRange(laptops, 5, 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>

// Define the structure to store attendance data
struct Student {
    int student_id;
    int total_classes;
    int classes_attended;
};

void calculate_attendance(struct Student student) {
    if (student.total_classes == 0) {
        printf("Total classes cannot be zero for student ID %d\n", student.student_id);
        return;
    }

    // Calculate the attendance percentage
    float attendance_percentage = ((float)student.classes_attended / student.total_classes) * 100;

    // Display the result
    printf("Student ID: %d\n", student.student_id);
    printf("Total Classes: %d\n", student.total_classes);
    printf("Classes Attended: %d\n", student.classes_attended);
    printf("Attendance Percentage: %.2f%%\n\n", attendance_percentage);
}

int main() {
    int n;
```

```
    // Ask the user for the number of students
    printf("Enter the number of students: ");
    scanf("%d", &n);

    // Create an array of students
    struct Student students[n];

    // Input attendance data for each student
    for (int i = 0; i < n; i++) {
        printf("\nEnter details for student %d\n", i + 1);
        printf("Student ID: ");
        scanf("%d", &students[i].student_id);
        printf("Total Classes: ");
        scanf("%d", &students[i].total_classes);
        printf("Classes Attended: ");
        scanf("%d", &students[i].classes_attended);
    }

    // Calculate and display attendance percentage for each student
    for (int i = 0; i < n; i++) {
        calculate_attendance(students[i]);
    }

    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>

// Define a structure for a flight
struct Flight {
    char flightNumber[10];
    char departure[50];
    char destination[50];
    int duration; // Duration in minutes
};

// Function to display flights that have a duration less than the specified limit
void displayShortFlights(struct Flight flights[], int numFlights, int maxDuration) {
    printf("Flights with a duration less than %d minutes:\n", maxDuration);
    for (int i = 0; i < numFlights; i++) {
        if (flights[i].duration < maxDuration) {
            printf("Flight 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\n", flights[i].duration);
        }
    }
}
```

```
int main() {
    // Example array of flights
    struct Flight flights[5] = {
        {"AA101", "New York", "London", 400},
        {"BB202", "Los Angeles", "Tokyo", 500},
        {"CC303", "Chicago", "Miami", 150},
        {"DD404", "San Francisco", "Dallas", 180},
        {"EE505", "Boston", "Washington", 60}
    };

    int numFlights = 5;
    int maxDuration;

    // Ask the user to specify the maximum duration
    printf("Enter the maximum flight duration (in minutes): ");
    scanf("%d", &maxDuration);

    // Display the flights that are shorter than the specified duration
    displayShortFlights(flights, numFlights, 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>
#include <stdlib.h>

// Define the structure to represent a polynomial term
struct Term {
    int coefficient;
    int exponent;
    struct Term* next; // Pointer to the next term
};

// Function to create a new term
struct Term* createTerm(int coeff, int exp) {
    struct Term* newTerm = (struct Term*)malloc(sizeof(struct Term));
    newTerm->coefficient = coeff;
    newTerm->exponent = exp;
    newTerm->next = NULL;
    return newTerm;
}

// Function to print the polynomial
void printPolynomial(struct Term* poly) {
    struct Term* temp = poly;
    while (temp != NULL) {
        if (temp->coefficient > 0 && temp != poly) {
            printf(" + ");
        }
        printf("%dx^%d", temp->coefficient, temp->exponent);
        temp = temp->next;
```

```
    }
    printf("\n");
}

// Function to add two polynomials
struct Term* addPolynomials(struct Term* poly1, struct Term* poly2) {
    struct Term* result = NULL;
    struct Term* last = NULL;

    // Traverse both polynomials
    while (poly1 != NULL && poly2 != NULL) {
        if (poly1->exponent == poly2->exponent) {
            int sumCoeff = poly1->coefficient + poly2->coefficient;
            if (sumCoeff != 0) {
                struct Term* newTerm = createTerm(sumCoeff, poly1->exponent);
                if (result == NULL) {
                    result = newTerm;
                } else {
                    last->next = newTerm;
                }
                last = newTerm;
            }
            poly1 = poly1->next;
            poly2 = poly2->next;
        } else if (poly1->exponent > poly2->exponent) {
            struct Term* newTerm = createTerm(poly1->coefficient, poly1->exponent);
            if (result == NULL) {
                result = newTerm;
            } else {
                last->next = newTerm;
            }
            last = newTerm;
            poly1 = poly1->next;
        } else {
            struct Term* newTerm = createTerm(poly2->coefficient, poly2->exponent);
            if (result == NULL) {
                result = newTerm;
            } else {
                last->next = newTerm;
            }
            last = newTerm;
            poly2 = poly2->next;
        }
    }

    // If there are any remaining terms in either polynomial
    while (poly1 != NULL) {
        struct Term* newTerm = createTerm(poly1->coefficient, poly1->exponent);
        if (result == NULL) {
            result = newTerm;
        } else {
            last->next = newTerm;
        }
        last = newTerm;
        poly1 = poly1->next;
```

```
    }

    while (poly2 != NULL) {
        struct Term* newTerm = createTerm(poly2->coefficient, poly2->exponent);
        if (result == NULL) {
            result = newTerm;
        } else {
            last->next = newTerm;
        }
        last = newTerm;
        poly2 = poly2->next;
    }

    return result;
}

// Function to multiply two polynomials
struct Term* multiplyPolynomials(struct Term* poly1, struct Term* poly2) {
    struct Term* result = NULL;
    struct Term* last = NULL;

    for (struct Term* temp1 = poly1; temp1 != NULL; temp1 = temp1->next) {
        for (struct Term* temp2 = poly2; temp2 != NULL; temp2 = temp2->next) {
            int newCoeff = temp1->coefficient * temp2->coefficient;
            int newExp = temp1->exponent + temp2->exponent;

            struct Term* newTerm = createTerm(newCoeff, newExp);

            // Insert the term into the result polynomial
            if (result == NULL) {
                result = newTerm;
            } else {
                last->next = newTerm;
            }
            last = newTerm;
        }
    }

    // Combine like terms (if any)
    struct Term* temp = result;
    struct Term* prev = NULL;

    while (temp != NULL && temp->next != NULL) {
        if (temp->exponent == temp->next->exponent) {
            temp->coefficient += temp->next->coefficient;
            struct Term* toDelete = temp->next;
            temp->next = temp->next->next;
            free(toDelete);
        } else {
            temp = temp->next;
        }
    }

    return result;
}
```

```
int main() {
    // Create the first polynomial: 3x^3 + 5x^2 + 2
    struct Term* poly1 = createTerm(3, 3);
    poly1->next = createTerm(5, 2);
    poly1->next->next = createTerm(2, 0);

    // Create the second polynomial: 4x^3 + 2x^2 + 1
    struct Term* poly2 = createTerm(4, 3);
    poly2->next = createTerm(2, 2);
    poly2->next->next = createTerm(1, 0);

    // Print both polynomials
    printf("Polynomial 1: ");
    printPolynomial(poly1);
    printf("Polynomial 2: ");
    printPolynomial(poly2);

    // Add the polynomials
    struct Term* sum = addPolynomials(poly1, poly2);
    printf("Sum of Polynomials: ");
    printPolynomial(sum);

    // Multiply the polynomials
    struct Term* product = multiplyPolynomials(poly1, poly2);
    printf("Product of Polynomials: ");
    printPolynomial(product);

    // Free the memory
    // In a real-world scenario, free the dynamically allocated memory here.

    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 a structure to represent the patient's medical record
struct Patient {
    char name[100];
    int age;
    char diagnosis[100];
    char treatment[100];
};

// Function to search patients by diagnosis
void searchByDiagnosis(struct Patient patients[], int numPatients, char diagnosis[]) {
    int found = 0;
    printf("Patients diagnosed with: %s\n", diagnosis);
    for (int i = 0; i < numPatients; i++) {
        if (strcmp(patients[i].diagnosis, diagnosis) == 0) {
```

```
            printf("\nName: %s\n", patients[i].name);
            printf("Age: %d\n", patients[i].age);
            printf("Diagnosis: %s\n", patients[i].diagnosis);
            printf("Treatment: %s\n", patients[i].treatment);
            found = 1;
        }
    }
    if (!found) {
        printf("No patients found with the diagnosis '%s'.\n", diagnosis);
    }
}

int main() {
    // Define an array of patients
    struct Patient patients[5] = {
        {"John Doe", 30, "Flu", "Rest and hydration"},
        {"Jane Smith", 45, "Diabetes", "Insulin therapy"},
        {"Alice Johnson", 29, "Flu", "Rest and hydration"},
        {"Bob Brown", 60, "Hypertension", "Blood pressure medication"},
        {"Carol White", 50, "Diabetes", "Insulin therapy"}
    };

    char searchDiagnosis[100];
    printf("Enter diagnosis to search for: ");
    fgets(searchDiagnosis, sizeof(searchDiagnosis), stdin);
    // Remove newline character
    searchDiagnosis[strcspn(searchDiagnosis, "\n")] = 0;

    // Call the search function
    searchByDiagnosis(patients, 5, searchDiagnosis);

    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>

// Define a structure to store player information
struct Player {
    char name[50];
    char game[50];
    int score;
};

// Function to find the top scorer for each game
void findTopScorer(struct Player players[], int n) {
    char games[10][50]; // Array to 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];

        // Loop through the players to find the top scorer for the game
        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 game %s: %s with score %d\n", games[i], topPlayer, maxScore);
    }
}

int main() {
    int n;

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

    struct Player players[n];

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

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

        printf("Enter the score of %s in %s: ", players[i].name, players[i].game);
        scanf("%d", &players[i].score);
    }

    // Display the top scorer for each game
    findTopScorer(players, n);

    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>

// Define the structure to store city information
struct City {
    char name[50];
    long population;
    float area;  // Area in square kilometers
};

// Function to calculate population density
float calculateDensity(struct City city) {
    return city.population / city.area;
}

int main() {
    // Declare an array to store multiple cities
    int n;

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

    // Declare an array of structures
    struct City cities[n];

    // Input city information
    for (int i = 0; i < n; i++) {
        printf("\nEnter details for city %d:\n", i + 1);
        printf("Name: ");
        getchar();  // Clear the buffer
        fgets(cities[i].name, 50, stdin);  // Read name
        // Remove newline character from the name
        cities[i].name[strcspn(cities[i].name, "\n")] = '\0';

        printf("Population: ");
        scanf("%ld", &cities[i].population);

        printf("Area (in square kilometers): ");
        scanf("%f", &cities[i].area);
    }

    // Calculate and display population density for each city
    printf("\nCity Population Density:\n");
    for (int i = 0; i < n; i++) {
        float density = calculateDensity(cities[i]);
        printf("City: %s\n", cities[i].name);
        printf("Population: %ld\n", cities[i].population);
        printf("Area: %.2f km²\n", cities[i].area);
        printf("Population Density: %.2f people per km²\n\n", density);
```

```
    }

    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>

// Define the structure for vehicle registration
struct Vehicle {
    char registration_number[15];
    char owner[50];
    char make[30];
    int year;
};

// Function to input vehicle details
void inputVehicleDetails(struct Vehicle *vehicle) {
    printf("\nEnter registration number: ");
    scanf("%s", vehicle->registration_number);

    printf("Enter owner's name: ");
    scanf("%s", vehicle->owner);

    printf("Enter vehicle make: ");
    scanf("%s", vehicle->make);

    printf("Enter the registration year: ");
    scanf("%d", &vehicle->year);
}

// Function to list all vehicles registered in a given year
void listVehiclesByYear(struct Vehicle vehicles[], int count, int year) {
    printf("\nVehicles registered in the year %d:\n", year);
    int found = 0;
    for (int i = 0; i < count; i++) {
        if (vehicles[i].year == year) {
            printf("\nRegistration Number: %s\n", vehicles[i].registration_number);
            printf("Owner: %s\n", vehicles[i].owner);
            printf("Make: %s\n", vehicles[i].make);
            printf("Year: %d\n", vehicles[i].year);
            found = 1;
        }
    }
    if (!found) {
        printf("No vehicles found for the year %d.\n", year);
    }
}

int main() {
    int n, search_year;
```

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

    // Declare an array of structures to store vehicle details
    struct Vehicle vehicles[n];

    // Input details of each vehicle
    for (int i = 0; i < n; i++) {
        printf("\nEnter details for vehicle %d:", i + 1);
        inputVehicleDetails(&vehicles[i]);
    }

    // Input the year to search for
    printf("\nEnter the year to search for: ");
    scanf("%d", &search_year);

    // List vehicles registered in the given year
    listVehiclesByYear(vehicles, n, search_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>

// Define the structure for a menu item
struct MenuItem {
    char name[50];
    char category[30];
    float price;
};

// Function to display menu items in a specific category
void displayMenuByCategory(struct MenuItem menu[], int size, char category[]) {
    printf("\nMenu items in the category: %s\n", category);
    printf("-------------------------------------\n");
    int found = 0;
    for (int i = 0; i < size; 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() {
```

```
    // Define a list of menu items
    struct MenuItem menu[] = {
        {"Burger", "Main Course", 5.99},
        {"Pasta", "Main Course", 7.49},
        {"Salad", "Appetizer", 3.99},
        {"Soup", "Appetizer", 2.99},
        {"Ice Cream", "Dessert", 4.50},
        {"Cake", "Dessert", 3.75},
        {"Coke", "Beverage", 1.50},
        {"Water", "Beverage", 0.99}
    };

    int menuSize = sizeof(menu) / sizeof(menu[0]);
    char category[30];

    // Ask user to input the category they want to see
    printf("Enter category (Main Course, Appetizer, Dessert, Beverage): ");
    scanf("%s", category);

    // Display menu items in the given category
    displayMenuByCategory(menu, menuSize, 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>

// Define a structure for a sports team
struct SportsTeam {
    char teamName[50];
    char sport[50];
    int numPlayers;
    char coach[50];
};

// Function to display all teams playing a specific sport
void displayTeamsBySport(struct SportsTeam teams[], int numTeams, char sport[]) {
    printf("Teams playing the sport: %s\n", sport);
    int found = 0;
    for (int i = 0; i < numTeams; i++) {
        if (strcmp(teams[i].sport, sport) == 0) {
            printf("Team Name: %s\n", teams[i].teamName);
            printf("Number of Players: %d\n", teams[i].numPlayers);
            printf("Coach: %s\n\n", teams[i].coach);
            found = 1;
        }
    }
    if (!found) {
        printf("No teams found for the sport: %s\n", sport);
    }
```

```
}

int main() {
    int numTeams = 3;  // Example number of teams

    // Define an array of sports teams
    struct SportsTeam teams[] = {
        {"The Falcons", "Soccer", 11, "John Doe"},
        {"Sharks United", "Basketball", 12, "Jane Smith"},
        {"Wildcats", "Soccer", 11, "Mark Johnson"}
    };

    char sportToSearch[50];
    printf("Enter the sport to search for teams: ");
    // Use scanf to input sport name
    scanf("%49s", sportToSearch);

    // Display teams playing the specific sport
    displayTeamsBySport(teams, numTeams, sportToSearch);

    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>

// Define a structure to store student details
struct Student {
    char name[50];
    int marks[5]; // assuming 5 subjects
    float total;
    float percentage;
};

// Function to calculate total and percentage
void calculateTotalAndPercentage(struct Student *student, int numSubjects) {
    student->total = 0;
    for (int i = 0; i < numSubjects; i++) {
        student->total += student->marks[i];
    }
    student->percentage = (student->total / (numSubjects * 100)) * 100; // assuming each subject is out of
100
}

int main() {
    int numStudents, numSubjects = 5;

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

    struct Student students[numStudents];
```

```
    // Loop through all students
    for (int i = 0; i < numStudents; i++) {
        // Input student name
        printf("\nEnter name of student %d: ", i + 1);
        scanf("%s", students[i].name);  // Read name using scanf

        // Input marks for each subject
        printf("Enter marks for 5 subjects (out of 100):\n");
        for (int j = 0; j < numSubjects; j++) {
            printf("Subject %d: ", j + 1);
            scanf("%d", &students[i].marks[j]);
        }

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

    // Output the results
    printf("\nStudent Results:\n");
    printf("Name\t\tTotal\tPercentage\n");
    for (int i = 0; i < numStudents; i++) {
        printf("%s\t%.2f\t%.2f%%\n", students[i].name, students[i].total, students[i].percentage);
    }

    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>

// Define a structure for the e-commerce product
struct Product {
    int product_id;
    char name[100];
    char category[50];
    float price;
    int stock;
};

// Function to update stock
void update_stock(struct Product *prod, int quantity) {
    prod->stock += quantity; // Update stock by adding the quantity
}

// Function to calculate total value of products in stock
float calculate_total_value(struct Product prod) {
    return prod.price * prod.stock; // Total value = price * stock
}

int main() {
    // Declare a product instance
```

```
    struct Product product;

    // Initialize product details
    printf("Enter product ID: ");
    scanf("%d", &product.product_id);

    printf("Enter product name: ");
    getchar();  // To consume the leftover newline character after the previous scanf
    fgets(product.name, sizeof(product.name), stdin);

    // Remove the trailing newline character if present
    product.name[strcspn(product.name, "\n")] = '\0';

    printf("Enter product category: ");
    fgets(product.category, sizeof(product.category), stdin);

    // Remove the trailing newline character if present
    product.category[strcspn(product.category, "\n")] = '\0';

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

    printf("Enter current stock quantity: ");
    scanf("%d", &product.stock);

    // Display product details
    printf("\nProduct details:\n");
    printf("ID: %d\n", product.product_id);
    printf("Name: %s\n", product.name);
    printf("Category: %s\n", product.category);
    printf("Price: %.2f\n", product.price);
    printf("Stock: %d\n", product.stock);

    // Update stock
    int quantity;
    printf("\nEnter the quantity to update stock (positive for adding, negative for removing): ");
    scanf("%d", &quantity);

    // Update stock based on user input
    update_stock(&product, quantity);

    // Display updated stock
    printf("Updated stock: %d\n", product.stock);

    // Calculate total value of products in stock
    float total_value = calculate_total_value(product);

    // Display total value
    printf("Total value of products in stock: %.2f\n", total_value);

    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>

// Structure to store album details
struct Album {
    char albumName[100];
    char artist[100];
    char genre[50];
    int releaseYear;
};

// Function to display albums of a specific genre
void displayAlbumsByGenre(struct Album albums[], int n, const char *genre) {
    int found = 0;
    for (int i = 0; i < n; i++) {
        if (strcmp(albums[i].genre, genre) == 0) {
            printf("\nAlbum Name: %s\n", albums[i].albumName);
            printf("Artist: %s\n", albums[i].artist);
            printf("Genre: %s\n", albums[i].genre);
            printf("Release Year: %d\n", albums[i].releaseYear);
            found = 1;
        }
    }

    if (!found) {
        printf("\nNo albums found for genre: %s\n", genre);
    }
}

int main() {
    // Defining an array of albums
    struct Album albums[] = {
        {"Album1", "Artist1", "Rock", 2020},
        {"Album2", "Artist2", "Pop", 2021},
        {"Album3", "Artist3", "Rock", 2019},
        {"Album4", "Artist4", "Jazz", 2022},
        {"Album5", "Artist5", "Pop", 2018}
    };

    int n = sizeof(albums) / sizeof(albums[0]); // Number of albums in the array

    // Input for the genre
    char genre[50];
    printf("Enter genre to search for albums: ");
    scanf("%s", genre); // Using scanf to read the genre input

    // Display albums of the given genre
    displayAlbumsByGenre(albums, n, 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>

// Structure to store ticket details
struct Ticket {
    char movieName[50];
    int seatNumber;
    float price;
};

int main() {
    int numTickets;
    float totalRevenue = 0.0;

    // Ask for the number of tickets to book
    printf("Enter the number of tickets to book: ");
    scanf("%d", &numTickets);

    // Declare an array of Ticket structures
    struct Ticket tickets[numTickets];

    // Input details for each ticket
    for (int i = 0; i < numTickets; i++) {
        printf("\nTicket %d\n", i + 1);

        // Movie name input (Note: %s reads only until the first space)
        printf("Enter movie name: ");
        scanf("%s", tickets[i].movieName);

        // Seat number input
        printf("Enter seat number: ");
        scanf("%d", &tickets[i].seatNumber);

        // Price input
        printf("Enter price of the ticket: ");
        scanf("%f", &tickets[i].price);

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

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

    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>
```

```c
#include <string.h>

// Define a structure to store course details
struct Course {
    char courseCode[10];
    char courseName[100];
    char instructor[100];
    int credits;
};

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

int main() {
    // Define an array of courses
    struct Course courses[5] = {
        {"CS101", "Introduction to Computer Science", "Dr. Smith", 3},
        {"CS102", "Data Structures", "Dr. Smith", 4},
        {"MATH101", "Calculus I", "Dr. Johnson", 3},
        {"CS201", "Algorithms", "Dr. Smith", 4},
        {"PHY101", "Physics I", "Dr. Lee", 3}
    };

    // Specify the instructor to search for
    char instructorName[100];
    printf("Enter instructor's name: ");
    // Using scanf to read the instructor name
    scanf("%99s", instructorName);  // "%99s" to avoid buffer overflow

    // Call the function to list courses by the specified instructor
    listCoursesByInstructor(courses, 5, instructorName);

    return 0;
}
```