Structures

```
#include <stdio.h>


// struct Date{
//     int date;
//     int months;
//     int year;
// };


// int main(){
//     struct Date today;
//     today.date = 8;
//     today.months=1;
//     today.year=2025;
//     printf( "Today date %d- %d -%d \n",today.date,today.months,today.year);



//     struct Date yesterday;
//     yesterday.date=7;
//     yesterday.months=1;
//     yesterday.year =2025;
//     printf("yesterday date %d- %d- %d
\n",yesterday.date,yesterday.months,yesterday.year);


//     // printf("Size of struct data %ld ",sizeof(struct Date));
//    printf("Size of struct data %ld \n",sizeof(today));
//    printf("Address of Today =  %p \n ",&today);
//    printf("Size of yesterday %ld \n",sizeof(yesterday));
//    printf("Address of yesterday %ld \n",&yesterday);
//    return 0;
// }



// #include <stdio.h>


// struct Date{
//    int date;
//    int months;
//    int year;
// };


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


//    printf( "Today date %d- %d -%d \n",today.date,today.months,today.year);



//    struct Date yesterday;
//    yesterday.date=7;
//    yesterday.months=1;
//    yesterday.year =2025;
//    printf("yesterday date %d- %d- %d
\n",yesterday.date,yesterday.months,yesterday.year);
```

```
//    return 0;

// }


// #include <stdio.h>


// struct Date{

//    int day;

//    int months;

//    int year;

// };


// int main(){

//    struct Date mydates[10];

//    mydates[3].months =1;

//    mydates[3].day =8;

//    mydates[3].year = 25;


//    printf( "Today date %d- %d -%d
\n",mydates[3].day,mydates[3].months,mydates[3].year);



//    return 0;

// }


// Nested structure

// #include <stdio.h>
// struct Date{
//    int day;
//    int months;
//    int year;
// };
// struct time
// {
//    int hour;
//    int minute;
//    int seconds;
// };

// struct date_time
// {
//    struct Date CurrentDate;
//    struct time CurrentTIme;
// };


// int main(){
//    struct date_time event;
//    event.CurrentDate.day =8;
//    event.CurrentTIme.hour = 12;
//    printf( "current Day %d \ncurrent Hour  -%d
\n",event.CurrentDate.day,event.CurrentTIme.hour);



//    return 0;
// }

// #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 \n",months[i].name,months[i].numberOfDays);
//     }
//     return 0;
// }




// // #include <stdio.h>
// // #include <string.h>
// // #define MAX 100

// // struct student
// // {
// //    char name[50];
// //    int rollNumber;
// //    float marks;
// // };
// // void addstudent(struct student students[], int *count);
// // void displayAllstudent(struct student students[], int count);
// // void findByRoll(struct student students[], int count);
// // void cal_aver(struct student students[], int count);

// // int main(){
// //    struct student students[MAX];
// //    int count =0;  // tracking the number of students
// //    int choice;
// //    while(1){
// //        printf("\nMenu:\n");
// //       printf("1. Add a New Student\n");
// //       printf("2. Display All Students\n");
// //       printf("3. Find and Display by Roll Number\n");
// //       printf("4. Calculate and Display Average Marks\n");
// //       printf("5. Exit\n");
// //       printf("Enter your choice: ");
// //       scanf("%d", &choice);

// //       switch(choice){
// //          case 1:
// //          addstudent(students,&count);
// //          break;
// //          case 2:
// //          displayAllstudent(students,count);
// //          break;
// //          case 3:
// //          findByRoll(students,count);
// //          break;
// //          case 4:
// //          cal_aver(students,count);
// //          break;
// //          case 5:
// //          printf("Exiting program.\n");
// //          return 0;
// //          default:
// //          printf("Invalid choice \n");

// //       }

// //    }
// //  return 0;

// // }

// // void addstudent(struct student students[], int *count){
// //    if(*count >MAX){
// //       printf("Cannot add more students \n");

// //    }
// //    printf("Enter Name \n");
// //    getchar();
// //    scanf("%s",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 displayAllstudent(struct student students[], int count){
// //    if (count == 0) {
// //        printf("No students available.\n");
// //        return;
// //    }
// //    for(int i=0;i<count;i++){
// //        printf("Student %d:\n", i + 1);
// //        printf("Name: %s\n", students[i].name);
// //        printf("Roll Number: %d\n", students[i].rollNumber);
// //        printf("Marks: %.2f\n", students[i].marks);

// //    }
// // }

// // void findByRoll(struct student students[], int count){
// //    if (count == 0) {
// //        printf("No students available.\n");
// //        return;
// //    }
// //    int rollNumber;
// //    printf("Enter roll number to search: ");
// //    scanf("%d", &rollNumber);
// //    for (int i = 0; i < count; i++) {
// //        if (students[i].rollNumber == rollNumber) {
// //            printf("Student Found:\n");
// //            printf("Name: %s\n", students[i].name);
// //            printf("Roll Number: %d\n", students[i].rollNumber);
// //            printf("Marks: %.2f\n", students[i].marks);
// //            return;
// //        }
// //        printf("Student not found ");
// //    }



// // }

// // void cal_aver(struct student students[], int count){
// //    if(count==0){
// //        printf("Student not there \n");
// //    }
// //    float total =0.0;
// //    for(int i=0;i<count;i++){
// //        total += students[i].marks;
// //    }
// //    float average = total/count;
// //    printf("Average of students marks is %.2f ",average);
// // }


// // Structure and pointer

// // #include <stdio.h>

// // struct Date{
// //    int day;
// //    int months;
// //    int year;
// // };

// // int main(){
// //    struct Date Today;
// //    struct Date *ptr;

// //    ptr = &Today;

// //    // (*ptr).months =1;
// //    ptr->months =1;
// //    printf("Today date : month =  %d \n",Today.months);
// //    return 0;
// // }


// // #include <stdio.h>
// // struct Date{
// //     int day;
// //     int months;
// //     int year;
// // };

// // int main(){
// //     struct Date today,*datePtr;

// //     datePtr = &today;

// //     datePtr->months =1;
// //     datePtr->day = 8;
// //     datePtr->year = 2025;

// //     printf("Today 's data is %d-%d-%d. \n",datePtr->months,datePtr->day,datePtr-
>year);
// //     return 0;
// // }

// // #include <stdio.h>

// struct intptrs{
//     int *p1;
//     int *p2;
// };

// int main(){
```

```
//    struct intptrs pointers;

//    int i1 =100,i2;

//    pointers.p1 = &i1;

//    pointers.p2 = &i2;

//    printf("i2 = %d, *pointer.p2 = %d \n",i2,*pointers.p2);


//    *pointers.p2 = -97;


//    printf("i1 = %d, *pointer.p1 = %d \n",i1,*pointers.p1);

//    printf("i2 = %d, *pointer.p2 = %d \n",i2,*pointers.p2);

// }



// #include <stdio.h>


// struct  num

// {

//    int a;

//    int 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("Sum = %d ",sumA);
```

```c
//    return 0;

// }

// int sum(struct num num1,struct num num2){

//    int sum = num1.a +num2.a;

//    return sum;


// }
```

Task 1

1) Student Information:

a. Define a structure to store student information, including name, roll number, and marks in three subjects.

b. Write a program to input data for 5 students and display the details along with their average marks.

```c
#include <stdio.h>


struct Student

{

   char name[50];

   int rollNumber;

   float marks[3];

};


int main(){

   struct Student student[5];

   for(int i=0;i<5;i++){
    printf("Enter the details of student %d",i+1);


    printf("\nEnter Name: ");
    scanf("%s",student[i].name);
    printf("Roll Number ");
    scanf("%d",&student[i].rollNumber);


    for(int j=0;j<3;j++){
        printf("Marks in subject %d: ", j + 1);
        scanf("%.2f \t", &student[i].marks[j]);
    }
    printf("\n");
}
printf("Students details and thier average marks ");
for(int i=0;i<5;i++){
    float total =0,average;
    printf("\nStudent %d:\n", i + 1);
    printf("Name: %s\n", student[i].name);
    printf("Roll Number: %d\n", student[i].rollNumber);
    printf("Marks: ");
    for(int j=0;j<3;j++){
        printf("%.2f",student[i].marks[j]);
        total += student[i].marks[j];
    }
    average  = total/3;
    printf("\n The average marks : %.2f\n",average);
}
return 0;
```

```
}
```

2) Employee Details:

a. Create a structure to store employee details like name, ID, salary, and department.

b. Write a function to display the details of employees whose salary is above a certain threshold.

```
#include <stdio.h>
#define THRESHOLD 50000

struct Employee
{
  char name[50];
  int ID;
  float salary;
  char department[50];
};

void displayHighSalary(struct Employee emp[]);
int main(){
struct Employee emp[3];
for(int i=0;i<3;i++){
  printf("Enter the details of employee : %d \n",i+1);

  printf("Enter Name: ");
  scanf(" %[^\n]s",emp[i].name);
  printf("Employee id: ");
  scanf("%d",&emp[i].ID);
```

```
    printf("Salary ");
    scanf("%f",&emp[i].salary);
    printf("Department ");
    scanf("%s",emp[i].department);
}
printf("\n");

    displayHighSalary(emp);
    return 0;
}
void displayHighSalary(struct Employee emp[]){
    printf("The details of employee whose salary is higher than threshold")
    for(int i=0;i<3;i++){
        if(emp[i].salary >THRESHOLD){
             printf("\nName: %s\n", emp[i].name);
            printf("ID: %d\n", emp[i].ID);
            printf("Salary: %.2f\n", emp[i].salary);
            printf("Department: %s\n", emp[i].department);
        }
    }
}
```

3) Book Store Inventory:

a. Define a structure to represent a book with fields for title, author, ISBN, and price.

b. Write a program to manage an inventory of books and allow searching by title.

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

```
#include <string.h>


struct Book

{

    char title[150];

    char author[100];

    char ISBN[20];

    float price;


};

void searchtitle(struct Book book[],char Title[]);


int main(){

    struct Book book[3];

    for(int i=0;i<3;i++){

        printf("Enter the details of book %d \n",i+1);


        printf("Enter title: ");

        scanf("%[^\n]s",book[i].title);

        getchar();

        printf("Enter Author: ");

        scanf(" %[^\n]s", book[i].author);

        getchar();

        printf("Enter ISBN: ");

        scanf("%s", book[i].ISBN);

        printf("Enter Price: ");

        scanf("%f", &book[i].price);

        getchar();
    }
    char Title[100];
    printf("\nEnter the title of the book to search: ");
    scanf(" %[^\n]s",Title);
    searchtitle(book,Title);
}
void searchtitle(struct Book book[],char Title[]){
    int found =0;
    for(int i=0;i<3;i++){
        if(strcmp(book[i].title,Title)==0){
            printf("\nBook found:\n");
            printf("Title: %s\n", book[i].title);
            printf("Author: %s\n", book[i].author);
            printf("ISBN: %s\n", book[i].ISBN);
            printf("Price: %.2f\n", book[i].price);
            found = 1;
            break;
        }

    }
    printf("\nBook not found");



}
```

4) Date Validation:

a. Create a structure to represent a date with day, month, and year.

b. Write a function to validate if a given date is correct (consider leap years).

```
#include <stdio.h>


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


int isValidDate(struct Date date) {
  int daysInMonth[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};


  if ((date.year % 4 == 0 && date.year % 100 != 0) || (date.year % 400 == 0)) {
    daysInMonth[1] = 29; // Leap year
  }


  if (date.year < 1 || date.month < 1 || date.month > 12 || date.day < 1 || date.day >
daysInMonth[date.month - 1]) {
    return 0; // Invalid date
  }
  return 1; // Valid date
}


int main() {
  struct Date dates[5];
  for (int i = 0; i < 5; i++) {
    printf("Enter date %d (DD MM YYYY): ", i + 1);
    scanf("%d %d %d", &dates[i].day, &dates[i].month, &dates[i].year);
    }

    printf("\nValidation Results:\n");
    for (int i = 0; i < 5; i++) {
        if (isValidDate(dates[i])) {
            printf("Date %d: %02d/%02d/%04d is valid.\n", i + 1, dates[i].day, dates[i].month,
dates[i].year);
        } else {
            printf("Date %d: %02d/%02d/%04d is invalid.\n", i + 1, dates[i].day,
dates[i].month, dates[i].year);
        }
    }
    return 0;
}
```

5) Complex Numbers:

a. Define a structure to represent a complex number with real and imaginary parts.

b. Implement functions to add, subtract, and multiply two complex numbers.

```
#include <stdio.h>
struct Complex
{
   float real;
   float imag;
};
struct Complex add(struct Complex a, struct Complex b) {
    struct Complex result;
    result.real = a.real + b.real;
    result.imag = a.imag + b.imag;
    return result;
}

struct Complex subtract(struct Complex a, struct Complex b) {
    struct Complex result;
    result.real = a.real - b.real;
    result.imag = a.imag - b.imag;
    return result;
}
struct Complex multiply(struct Complex a, struct Complex b) {
    struct Complex result;
    result.real = a.real * b.real - a.imag * b.imag;
    result.imag = a.real * b.imag + a.imag * b.real;
    return result;
}

int main(){
    struct Complex num1,num2,result;
    printf("Enter the real and imaginary parts of the first complex number: ");
    scanf("%f %f", &num1.real, &num1.imag);
    printf("Enter the real and imaginary parts of the second complex number: ");
    scanf("%f %f", &num2.real, &num2.imag);

    result = add(num1, num2);
    printf("\nAddition: %.2f + %.2fi\n", result.real, result.imag);

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

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

    return 0;
}
```

6) Bank Account:

a. Design a structure to store information about a bank account, including account number, account holder name, and balance.

b. Write a function to deposit and withdraw money, and display the updated balance.

```
#include <stdio.h>

struct BankAccount {
    int accountNumber;
    char accountholder[100];
    float balance;
};

void deposit(struct BankAccount *account, float amount);
void withdraw(struct BankAccount *account, float amount);

int main() {
    struct BankAccount account;
```

```
printf("Enter the account number: ");

scanf("%d", &account.accountNumber);

printf("Enter account holder name: ");

scanf(" %[^\n]s", account.accountholder);

printf("Enter initial balance: ");

scanf("%f", &account.balance);


int choice;

float amount;


do {

  printf("\n--- Bank Operations ---");

  printf("\n1. Deposit Money");

  printf("\n2. Withdraw Money");

  printf("\n3. Display Balance");

  printf("\n4. Exit");

  printf("\nEnter your choice: ");

  scanf("%d", &choice);


  switch (choice) {

    case 1:

      printf("\nEnter the amount to deposit: ");

      scanf("%f", &amount);

      deposit(&account, amount);

      break;


    case 2:

      printf("\nEnter the amount to withdraw: ");
```

```
            scanf("%f", &amount);
            withdraw(&account, amount);
            break;

          case 3:

            printf("\nAccount Number: %d", account.accountNumber);
            printf("\nAccount Holder: %s", account.accountholder);
            printf("\nCurrent Balance: %.2f\n", account.balance);
            break;

          case 4:
            printf("\nExiting... Thank you!\n");
            break;

          default:
            printf("\nInvalid choice! Please try again.\n");
        }
    } while (choice != 4);

    return 0;
}

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

void withdraw(struct BankAccount *account, float amount) {
    if (amount > 0 && amount <= account->balance) {
        account->balance -= amount;
        printf("\n%.2f withdrawn successfully. Updated balance: %.2f\n", amount, account-
>balance);
    } else if (amount > account->balance) {
        printf("\nInsufficient balance!\n");
    } else {
        printf("\nInvalid withdrawal amount!\n");
    }
}
```

7) Car Inventory System:

a. Create a structure for a car with fields like make, model, year, and price.

b. Write a program to store details of multiple cars and print cars within a specified price range.

```
#include <stdio.h>

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

int main() {
    struct Car cars[3];
    for (int i = 0; i < 3; i++) {
        scanf("%s %s %d %f", cars[i].make, cars[i].model, &cars[i].year, &cars[i].price);
    }
    float minPrice, maxPrice;
    scanf("%f %f", &minPrice, &maxPrice);
    for (int i = 0; i < 3; i++) {
        if (cars[i].price >= minPrice && cars[i].price <= maxPrice) {
            printf("%s %s %d %.2f\n", cars[i].make, cars[i].model, cars[i].year, cars[i].price);
        }
    }
    return 0;
}
```

8) Library Management:

a. Define a structure for a library book with fields for title, author, publication year, and status (issued or available).

b. Write a function to issue and return books based on their status.

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

```c
struct Book {
    char title[50];
    char author[50];
    int year;
    int status; // 0 = available, 1 = issued
};

void issue(struct Book *book) {
    if (book->status == 0) {
        book->status = 1;
        printf("Book issued\n");
    } else {
        printf("Book already issued\n");
    }
}

void returnBook(struct Book *book) {
    if (book->status == 1) {
        book->status = 0;
        printf("Book returned\n");
    } else {
        printf("Book already available\n");
    }
}

int main() {
    struct Book books[2] = {{"Book1", "Author1", 2000, 0}, {"Book2", "Author2", 2010, 1}};
    issue(&books[0]);
    returnBook(&books[1]);
    return 0;
}
```

9) Student Grades:

a. Create a structure to store a student's name, roll number, and an array of grades.

b. Write a program to calculate and display the highest, lowest, and average grade for each student.

```
#include <stdio.h>

struct Student {
    char name[50];
    int roll;
    float grades[3];
};

int main() {
    struct Student students[2];
    for (int i = 0; i < 2; i++) {
        scanf("%s %d", students[i].name, &students[i].roll);
        for (int j = 0; j < 3; j++) {
            scanf("%f", &students[i].grades[j]);
        }
    }
    for (int i = 0; i < 2; i++) {
        float max = students[i].grades[0], min = students[i].grades[0], sum = 0;
    for (int j = 0; j < 3; j++) {
      if (students[i].grades[j] > max) max = students[i].grades[j];
      if (students[i].grades[j] < min) min = students[i].grades[j];
      sum += students[i].grades[j];
    }
    printf("%s %d Max: %.2f Min: %.2f Avg: %.2f\n", students[i].name, students[i].roll,
max, min, sum / 3);
  }
  return 0;
}
```

10) Product Catalog:

a. Define a structure to represent a product with fields for product ID, name, quantity, and price.

b. Write a program to update the quantity of products after a sale and calculate the total sales value.

```
#include <stdio.h>

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

int main() {
  struct Product products[3];
  for (int i = 0; i < 3; i++) {
    scanf("%d %s %d %f", &products[i].id, products[i].name, &products[i].quantity,
&products[i].price);
  }
  int saleID, saleQty;
  scanf("%d %d", &saleID, &saleQty);
  for (int i = 0; i < 3; i++) {
    if (products[i].id == saleID && saleQty <= products[i].quantity) {
      products[i].quantity -= saleQty;
      printf("Updated Quantity: %d Total Sale: %.2f\n", products[i].quantity, saleQty *
products[i].price);
    }
  }
  return 0;
}
```

Task 2

1. Point Distance Calculation:

o Define a structure for a point in 2D space (x, y).

o Write a function to calculate the distance between two points.

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

struct Distance
{
  int x;
```

```
    int y;
};

float distance(struct Distance x,struct Distance y);

int main(){
    struct Distance x1,y1;
    printf("Enter points \n");
    scanf("%d %d",&x1.x,&x1.y);
    scanf("%d %d",&y1.x,&y1.y);
    float point_distance = distance(x1,y1);
    printf("The distance between two point is %.2f",point_distance);
}

float distance(struct Distance x1,struct Distance y1){
    float point_distance = sqrt(pow(y1.x-x1.x,2)+pow(y1.y-x1.y,2));
    return point_distance;
}
```

2. Rectangle Properties:

o Create a structure for a rectangle with length and width.

o Write functions to calculate the area and perimeter of the rectangle.

```
#include <stdio.h>

struct rectangle{
    int length;
    int width;
};
int area(struct rectangle a);
int perimeter(struct rectangle p);

int main(){
    struct rectangle rect;
    printf("Enter length and width of the rectangle: ");
    scanf("%d %d",&rect.length,&rect.width);
    printf("Area: %d\n", area(rect));
    printf("Perimeter: %d \n", perimeter(rect));

    return 0;

}
int area(struct rectangle a){
    return a.length * a.width;
}
int perimeter(struct rectangle p){
    return 2*(p.length+p.width);
}
```

3. Movie Details:

o Define a structure to store details of a movie, including title, director, release year, and rating.

o Write a program to sort movies by their rating.

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

```
#include<string.h>


struct Movie
{
  char title[100];
  char director[100];
  int realseyear;
  float rating;
};
void sortMovie(struct Movie movies[]);
int main(){
   struct Movie movies[3];


   for(int i=0;i<3;i++){
      printf("Enter details for movie %d\n", i + 1);
      printf("Title: ");
      scanf(" %[^\n]s", movies[i].title);
      printf("Director: ");
      scanf(" %[^\n]s", movies[i].director);
      printf("Release Year: ");
      scanf("%d", &movies[i].realseyear);
      printf("Rating: ");
      scanf("%f", &movies[i].rating);


   }
   sortMovie(movies);
   printf("Movies are sorted by rating: \n");
   for (int i = 0; i < 3; i++) {
        printf("\nMovie %d\n", i + 1);

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

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

        printf("Release Year: %d\n", movies[i].realseyear);

        printf("Rating: %.1f\n", movies[i].rating);

    }


    return 0;

}

void sortMovie(struct Movie movies[]){

    struct Movie temp;

    for(int i=0;i<3-1;i++){

        for(int j=i+1;j<3;j++){

            if(movies[i].rating < movies[j].rating){

                temp = movies[i];

                movies[i] = movies[j];

                movies[j] = temp;

            }

        }

    }

}
```

4. Weather Report:

o Create a structure to store daily weather data, including date, temperature, and humidity.

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

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

```
struct weather{

   int date;

   float temperatur;

   float humidity;

};


int main(){

   struct weather data[5];

   int maxTempDate =0;

   float max_temp =0;

   for(int i=0;i<5;i++){

      printf(" enter the weather report of Details %d \n",i+1);

      printf("Date (DDMMYYYY): ");

      scanf("%d",&data[i].date);

      printf("Temperatur :");

      scanf("%f",&data[i].temperatur);

      printf("Humidity : ");

      scanf("%f",&data[i].humidity);


       if(data[i].temperatur>max_temp){

         max_temp = data[i].temperatur;

         maxTempDate = data[i].date;


      }

   }

   printf("\nThe highest temperature was %.2f on date %d.\n", max_temp,
maxTempDate);
```

```
    return 0;

}
```

5. Fraction Arithmetic:

o Define a structure for a fraction with numerator and denominator.

o Write functions to add, subtract, multiply, and divide two fractions.

```
#include <stdio.h>

struct Fraction {
    int numerator;
    int denominator;
};

struct Fraction add(struct Fraction f1, struct Fraction f2);
struct Fraction subtract(struct Fraction f1, struct Fraction f2);
struct Fraction multiply(struct Fraction f1, struct Fraction f2);
struct Fraction divide(struct Fraction f1, struct Fraction f2);
void simplify(struct Fraction *f);

int main() {
    struct Fraction f1, f2, result;
    printf("Enter the first fraction (numerator denominator): ");
    scanf("%d %d", &f1.numerator, &f1.denominator);
    printf("Enter the second fraction (numerator denominator): ");
    scanf("%d %d", &f2.numerator, &f2.denominator);
    result = add(f1, f2);
    simplify(&result);
    printf("\nAddition: %d/%d", result.numerator, result.denominator);

    result = subtract(f1, f2);
    simplify(&result);
    printf("\nSubtraction: %d/%d", result.numerator, result.denominator);

    result = multiply(f1, f2);
    simplify(&result);
    printf("\nMultiplication: %d/%d", result.numerator, result.denominator);

    result = divide(f1, f2);
    simplify(&result);
    printf("\nDivision: %d/%d\n", result.numerator, result.denominator);

    return 0;
}

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

struct Fraction subtract(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;
}

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

struct Fraction divide(struct Fraction f1, struct Fraction f2) {
    struct Fraction result;
    result.numerator = f1.numerator * f2.denominator;
    result.denominator = f1.denominator * f2.numerator;
    return result;
}

void simplify(struct Fraction *f) {
    int a = f->numerator, b = f->denominator, gcd;
    while (b != 0) {
        gcd = b;
        b = a % b;
        a = gcd;
    }
    f->numerator /= gcd;
    f->denominator /= gcd;
    if (f->denominator < 0) { // Ensure denominator is positive
        f->numerator = -f->numerator;
        f->denominator = -f->denominator;
    }
}
```

6. Laptop Inventory:

o Create a structure to represent a laptop with fields for brand, model, processor, RAM, and price.

o Write a program to list laptops within a specific price range.

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

struct Laptop {
    char brand[50];
    char model[50];
    char processor[50];
    int ram;      // RAM in GB
    float price;   // Price in INR
};

void listLaptopsInRange(struct Laptop laptops[], int n, float minPrice, float maxPrice);

int main() {
    int n;
    printf("Enter the number of laptops: ");
    scanf("%d", &n);

    struct Laptop laptops[n];

    // Input laptop details
    for (int i = 0; i < n; i++) {
        printf("\nEnter details for Laptop %d:\n", i + 1);
        printf("Brand: ");
        scanf("%s", laptops[i].brand);
        printf("Model: ");
        scanf("%s", laptops[i].model);
        printf("Processor: ");
        scanf("%s", laptops[i].processor);
        printf("RAM (in GB): ");
        scanf("%d", &laptops[i].ram);
        printf("Price (in INR): ");
        scanf("%f", &laptops[i].price);
    }

    float minPrice, maxPrice;
    printf("\nEnter the price range (min max): ");
    scanf("%f %f", &minPrice, &maxPrice);

    // List laptops within the specified price range
    listLaptopsInRange(laptops, n, minPrice, maxPrice);

    return 0;
}
```

```
void listLaptopsInRange(struct Laptop laptops[], int n, float minPrice, float maxPrice) {
  printf("\nLaptops in the price range %.2f - %.2f:\n", minPrice, maxPrice);
  int found = 0;
  for (int i = 0; i < n; i++) {
    if (laptops[i].price >= minPrice && laptops[i].price <= maxPrice) {
      printf("\nBrand: %s\nModel: %s\nProcessor: %s\nRAM: %dGB\nPrice: %.2f
INR\n",
          laptops[i].brand, laptops[i].model, laptops[i].processor, laptops[i].ram,
laptops[i].price);
      found = 1;
    }
  }
  if (!found) {
    printf("\nNo laptops found in the specified price range.\n");
  }
}
```

7. Student Attendance:

o Define a structure to store attendance data, including student ID, total classes, and classes attended.

o Write a program to calculate and display the attendance percentage for each student.

```
#include <stdio.h>

struct Attendance {
  int studentID;
```

```
    int totalClasses;

    int classesAttended;

};


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


int main() {

    int n;

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

    scanf("%d", &n);


    struct Attendance students[n];


    for (int i = 0; i < n; 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);

    }


    calculateAttendance(students, n);


    return 0;

}
```

```
void calculateAttendance(struct Attendance students[], int n) {
  printf("\n--- Attendance Report ---\n");
  for (int i = 0; i < n; i++) {
    float percentage = ((float)students[i].classesAttended / students[i].totalClasses) *
100;
    printf("\nStudent 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", percentage);
  }
}
```

8. Flight Information:

o Create a structure for a flight with fields for flight number, departure, destination, and duration.

o Write a program to display flights that are less than a specified duration.

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

struct Flight {
  int flightNumber;
  char departure[50];
  char destination[50];
  float duration;
};

int main() {
  int n, threshold;

  printf("Enter the number of flights: ");

  scanf("%d", &n);

  struct Flight flights[n];


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

      printf("\nEnter details for Flight %d:\n", i + 1);

      printf("Flight Number: ");

      scanf("%d", &flights[i].flightNumber);

      printf("Departure: ");

      scanf("%s", flights[i].departure);

      printf("Destination: ");

      scanf("%s", flights[i].destination);

      printf("Duration (hours): ");

      scanf("%f", &flights[i].duration);

  }


  printf("\nEnter the maximum flight duration: ");

  scanf("%f", &threshold);


  printf("\n--- Flights with duration less than %.2f hours ---\n", threshold);

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

      if (flights[i].duration < threshold) {

          printf("Flight %d: %s to %s (%.2f hours)\n", flights[i].flightNumber,
flights[i].departure, flights[i].destination, flights[i].duration);

      }

  }
  return 0;
}
```

9. Polynomial Representation:

o Define a structure to represent a term of a polynomial (coefficient and exponent).

o Write functions to add and multiply two polynomials.

```
#include <stdio.h>

struct Term {
    int coefficient;
    int exponent;
};

void addPolynomials(struct Term p1[], int n1, struct Term p2[], int n2, struct Term
result[], int *size);
void multiplyPolynomials(struct Term p1[], int n1, struct Term p2[], int n2, struct Term
result[], int *size);

int main() {
    struct Term p1[] = {{3, 2}, {5, 1}, {6, 0}};
    struct Term p2[] = {{4, 1}, {2, 0}};
    struct Term result[10];
    int size;

    addPolynomials(p1, 3, p2, 2, result, &size);
    printf("\nSum of Polynomials:\n");
    for (int i = 0; i < size; i++) {
        printf("%dx^%d ", result[i].coefficient, result[i].exponent);
    }

    multiplyPolynomials(p1, 3, p2, 2, result, &size);
    printf("\n\nProduct of Polynomials:\n");
    for (int i = 0; i < size; i++) {
        printf("%dx^%d ", result[i].coefficient, result[i].exponent);
    }

    return 0;
}
```

Functions omitted for brevity

10. Medical Records:

o Create a structure for a patient's medical record with fields for name, age, diagnosis, and treatment.

o Write a program to search for patients by diagnosis.

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

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

```
};

int main() {
  int n;
  printf("Enter the number of patients: ");
  scanf("%d", &n);

  struct MedicalRecord records[n];

  for (int i = 0; i < n; 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);
  }

  char searchDiagnosis[100];
  printf("\nEnter the diagnosis to search for: ");
  scanf(" %[^\n]", searchDiagnosis);

  // Search and display patients with the given diagnosis
  printf("\n--- Patients with Diagnosis: %s ---\n", searchDiagnosis);
  int found = 0;  // Flag to track if any patients were found
```

```
    for (int i = 0; i < n; i++) {
        if (strcmp(records[i].diagnosis, searchDiagnosis) == 0) {
            printf("Name: %s, Age: %d, Treatment: %s\n", records[i].name, records[i].age,
records[i].treatment);
            found = 1;
        }
    }

    if (!found) {
        printf("No patients found with the diagnosis '%s'.\n", searchDiagnosis);
    }

    return 0;
}
```

11. Game Scores:

o Define a structure to store player information, including name, game played, and score.

o Write a program to display the top scorer for each game.

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

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

```
int main() {
  int n;
  printf("Enter the number of players: ");
  scanf("%d", &n);

  struct Player players[n];
  for (int i = 0; i < n; 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);
  }

  printf("\n--- Top Scorers for Each Game ---\n");
  for (int i = 0; i < n; i++) {
    int maxScore = players[i].score;
    int maxIndex = i;
    int alreadyProcessed = 0;
    for (int j = 0; j < i; j++) {
      if (strcmp(players[i].game, players[j].game) == 0) {
        alreadyProcessed = 1;
        break;
      }
    }
```

```
        if (alreadyProcessed) {
            continue;
        }
        for (int j = 0; j < n; j++) {
            if (strcmp(players[i].game, players[j].game) == 0 && players[j].score > maxScore) {
                maxScore = players[j].score;
                maxIndex = j;
            }
        }

        printf("Game: %s\n", players[maxIndex].game);
        printf("Top Scorer: %s, Score: %d\n\n", players[maxIndex].name,
players[maxIndex].score);
    }

    return 0;
}
```

12. City Information:

o Create a structure to store information about a city, including name, population, and area.

o Write a program to calculate and display the population density of each city.

```
#include <stdio.h>

struct City{
    char name[100];
    int population;
    float area;
```

```
};

int main(){

    int n;

    printf("Enter the number of cities: ");

    scanf("%d", &n);

    struct City city[n];

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

        printf("\nEnter the details for city %d \n",i+1);

        printf("Name ");

        getchar();

        scanf("%[^\n]s",city[i].name);

        printf("Population : ");

        scanf("%d",&city[i].population);

        printf("Area (in square kilometers): ");

        scanf("%f", &city[i].area);

        getchar();

    }

    printf("\n-----------City Info and population Density-----------\n");


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

        float density = city[i].population/city[i].area;

        printf("City: %s\n", city[i].name);

        printf("Population: %d\n", city[i].population);

        printf("Area: %.2f sq.km\n", city[i].area);

        printf("Population Density: %.2f people/sq.km\n\n", density);

    }

    return 0;
```

```
}
```

13. Vehicle Registration:

o Define a structure for vehicle registration details, including registration number, owner, make, and year.

o Write a program to list all vehicles registered in a given year.

```
#include <stdio.h>

#include <string.h>


struct Vehicle {

    char registration_number[20];

    char owner[50];

    char make[30];

    int year;

};


int main() {

    int n, search_year;

    printf("Enter the number of vehicles: ");

    scanf("%d", &n);


    struct Vehicle vehicles[n];    // Input vehicle details

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

        printf("\nEnter details for Vehicle %d:\n", i + 1);

        printf("Registration Number: ");

        scanf(" %[^\n]", vehicles[i].registration_number);
        printf("Owner: ");

        scanf(" %[^\n]", vehicles[i].owner);

        printf("Make: ");

        scanf(" %[^\n]", vehicles[i].make);

        printf("Year of Registration: ");

        scanf("%d", &vehicles[i].year);

    }


    printf("\nEnter the year to search for registered vehicles: ");

    scanf("%d", &search_year);


    printf("\n--- Vehicles Registered in Year %d ---\n", search_year);

    int found = 0;

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

        if (vehicles[i].year == search_year) {

            printf("\nVehicle %d Details:\n", i + 1);

            printf("Registration Number: %s\n", vehicles[i].registration_number);

            printf("Owner: %s\n", vehicles[i].owner);

            printf("Make: %s\n", vehicles[i].make);

            printf("Year of Registration: %d\n", vehicles[i].year);

            found = 1;

        }

    }


    if (!found) {

        printf("\nNo vehicles found registered in the year %d.\n", search_year);

    }
```

```
    return 0;
}
```

14. Restaurant Menu:

o Create a structure to represent a menu item with fields for name, category, and price.

o Write a program to display menu items in a specific category.

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

struct MenuItem {
    char name[50];
    char category[30];
    float price;
};

void displayMenuByCategory(struct MenuItem menu[], int size, const char *category);

int main() {
    int n;
    printf("Enter the number of menu items: ");
    scanf("%d", &n);

    struct MenuItem menu[n];
    for (int i = 0; i < n; 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("%f", &menu[i].price);

    }


    char search_category[30];

    printf("\nEnter the category to display menu items: ");

    scanf(" %[^\n]", search_category);


    displayMenuByCategory(menu, n, search_category);


    return 0;

}


void displayMenuByCategory(struct MenuItem menu[], int size, const char *category) {

    int found = 0;

    printf("\n--- Menu Items in Category: %s ---\n", category);

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

        if (strcmp(menu[i].category, category) == 0) {

            printf("\nItem: %s\n", menu[i].name);

            printf("Price: %.2f\n", menu[i].price);

            found = 1;

        }

    }


    if (!found) {
        printf("\nNo menu items found in the '%s' category.\n", category);
    }
}
```

15. Sports Team:

o Define a structure for a sports team with fields for team name, sport, number of players, and coach.

o Write a program to display all teams playing a specific sport.

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

struct SportsTeam {
    char teamName[50];
    char sport[30];
    int numberOfPlayers;
    char coach[50];
};

void displayTeamsBySport(struct SportsTeam teams[], int size, const char *sport);

int main() {
    int n;

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

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

    char searchSport[30];
    printf("\nEnter the sport to display teams: ");
    scanf(" %[^\n]", searchSport);

    displayTeamsBySport(teams, n, searchSport);

    return 0;
}

void displayTeamsBySport(struct SportsTeam teams[], int size, const char *sport) {
    int found = 0;
    printf("\n--- Teams playing %s ---\n", sport);
    for (int i = 0; i < size; i++) {
        if (strcmp(teams[i].sport, sport) == 0) {
            printf("\nTeam: %s\n", teams[i].teamName);
            printf("Coach: %s\n", teams[i].coach);
            printf("Number of Players: %d\n", teams[i].numberOfPlayers);
            found = 1;
        }
    }

    if (!found) {
        printf("\nNo teams found playing %s.\n", sport);
    }
}
```

16. Student Marks Analysis:

- Create a structure to store student marks in different subjects.
- Write a program to calculate the total and percentage of marks for each student.

```
#include <stdio.h>

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

void calculateTotalAndPercentage(struct Student *s);

int main() {
    int n;
    printf("Enter number of students: ");
    scanf("%d", &n);

    struct Student students[n];

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

        calculateTotalAndPercentage(&students[i]);
    }

    printf("\nStudent Marks Analysis:\n");
    for (int i = 0; i < n; i++) {
        printf("\nStudent: %s\n", students[i].name);
        printf("Total Marks: %d\n", students[i].total);
        printf("Percentage: %.2f%%\n", students[i].percentage);
    }

    return 0;
}
```

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

17. E-commerce Product:

o Define a structure for an e-commerce product with fields for product ID, name, category, price, and stock.

o Write a program to update the stock and calculate the total value of products in stock.

```
#include <stdio.h>

struct Product {
    int productID;
    char name[50];
    char category[30];
    float price;
    int stock;
};

void updateStockAndCalculateValue(struct Product *p);

int main() {
    int n;
    printf("Enter the number of products: ");
    scanf("%d", &n);

    struct Product products[n];

    for (int i = 0; i < n; i++) {
        printf("\nEnter details for 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("%f", &products[i].price);
        printf("Stock: ");
        scanf("%d", &products[i].stock);

        updateStockAndCalculateValue(&products[i]);
    }

    return 0;
}

void updateStockAndCalculateValue(struct Product *p) {
    int sold;

    printf("\nEnter the number of products sold for %s: ", p->name);
    scanf("%d", &sold);
    p->stock -= sold;

    float totalValue = p->stock * p->price;
    printf("Updated stock for %s: %d\n", p->name, p->stock);
    printf("Total value of remaining stock: %.2f\n", totalValue);
}
```

18. Music Album:

o Create a structure to store details of a music album, including album name, artist, genre, and release year.

o Write a program to display albums of a specific genre.

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

struct MusicAlbum {
    char albumName[50];
    char artist[50];
    char genre[30];
    int releaseYear;
};

void displayAlbumsByGenre(struct MusicAlbum albums[], int size, const char *genre);

int main() {
    int n;
    printf("Enter the number of albums: ");
    scanf("%d", &n);

    struct MusicAlbum albums[n];

    for (int i = 0; i < n; 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);
    }

    char searchGenre[30];
    printf("\nEnter the genre to display albums: ");
    scanf(" %[^\n]", searchGenre);

    displayAlbumsByGenre(albums, n, searchGenre);

    return 0;
}

void displayAlbumsByGenre(struct MusicAlbum albums[], int size, const char *genre) {
    int found = 0;
    printf("\n--- Albums of Genre: %s ---\n", genre);
    for (int i = 0; i < size; i++) {
        if (strcmp(albums[i].genre, genre) == 0) {
            printf("\nAlbum: %s\n", albums[i].albumName);
            printf("Artist: %s\n", albums[i].artist);
            printf("Release Year: %d\n", albums[i].releaseYear);
            found = 1;
        }
    }

    if (!found) {
        printf("\nNo albums found for the genre '%s'.\n", genre);
    }
}
```

19. Cinema Ticket Booking:

o Define a structure for a cinema ticket with fields for movie name, seat number, and price.

o Write a program to book tickets and display the total revenue generated.

```
#include <stdio.h>

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

```
void bookTicket(struct Ticket *t);
float calculateRevenue(struct Ticket tickets[], int size);

int main() {
  int n;

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

  struct Ticket tickets[n];

  for (int i = 0; i < n; i++) {
    printf("\nEnter details for Ticket %d:\n", i + 1);
    bookTicket(&tickets[i]);
  }

  float revenue = calculateRevenue(tickets, n);
  printf("\nTotal revenue generated: %.2f\n", revenue);

  return 0;
}

void bookTicket(struct Ticket *t) {
  printf("Movie Name: ");
  scanf(" %[^\n]", t->movieName);
  printf("Seat Number: ");
  scanf("%d", &t->seatNumber);
  printf("Ticket Price: ");
```

```
    scanf("%f", &t->price);
}

float calculateRevenue(struct Ticket tickets[], int size) {
    float totalRevenue = 0;
    for (int i = 0; i < size; i++) {
        totalRevenue += tickets[i].price;
    }
    return totalRevenue;
}
```

20. University Courses:

o Create a structure to store course details, including course code, name, instructor, and credits.

o Write a program to list all courses taught by a specific instructor.

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

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

void listCoursesByInstructor(struct Course courses[], int size, const char *instructor);
```

```
int main() {
    int n;

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

    struct Course courses[n];

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

    // Get the instructor name to search for courses
    char searchInstructor[50];
    printf("\nEnter the instructor's name to list their courses: ");
    scanf(" %[^\n]", searchInstructor);
    // List all courses taught by the specified instructor
    listCoursesByInstructor(courses, n, searchInstructor);

    return 0;
}

// Function to list all courses taught by a specific instructor
void listCoursesByInstructor(struct Course courses[], int size, const char *instructor) {
    int found = 0;
    printf("\n--- Courses taught by %s ---\n", instructor);

    for (int i = 0; i < size; i++) {
        if (strcmp(courses[i].instructor, instructor) == 0) {
            printf("\nCourse Code: %s\n", courses[i].courseCode);
            printf("Course Name: %s\n", courses[i].name);
            printf("Credits: %d\n", courses[i].credits);
            found = 1;
        }
    }

    if (!found) {
        printf("\nNo courses found for the instructor '%s'.\n", instructor);
    }
}
```