DAY 14 -DAILY ASSIGNMENTS

ANSU MARIUM SHIBU

25-11-2024

1.Problem 1: Book Inventory System

Problem Statement:

Write a C program to manage a book inventory system using dynamic memory allocation. The program should:

Define a structure named Book with the following fields:

id (integer): The book's unique identifier.

title (character array of size 100): The book's title.

price (float): The price of the book.

Dynamically allocate memory for n books (where n is input by the user).

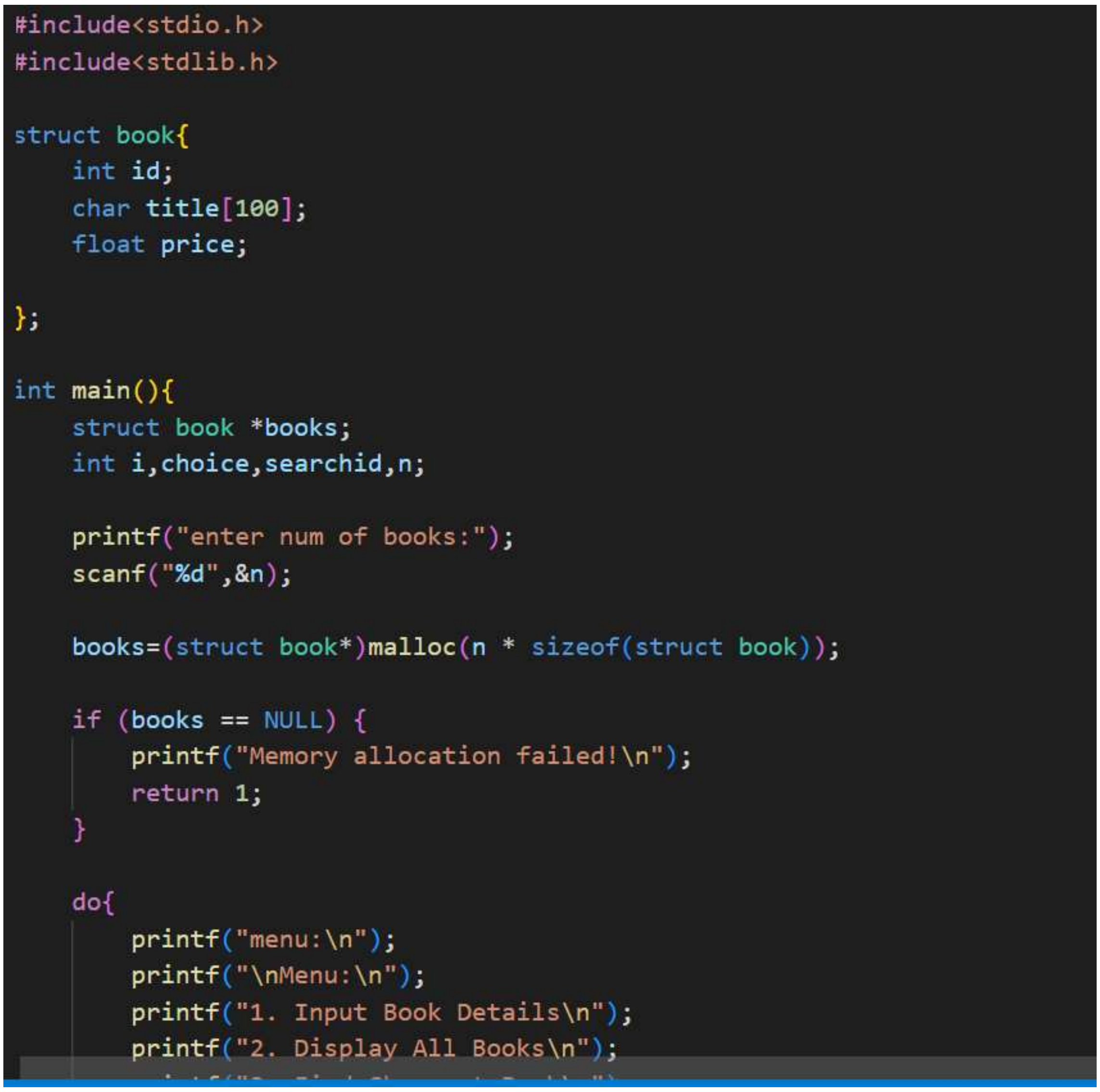
Implement the following features:

Input Details: Input details for each book (ID, title, and price).

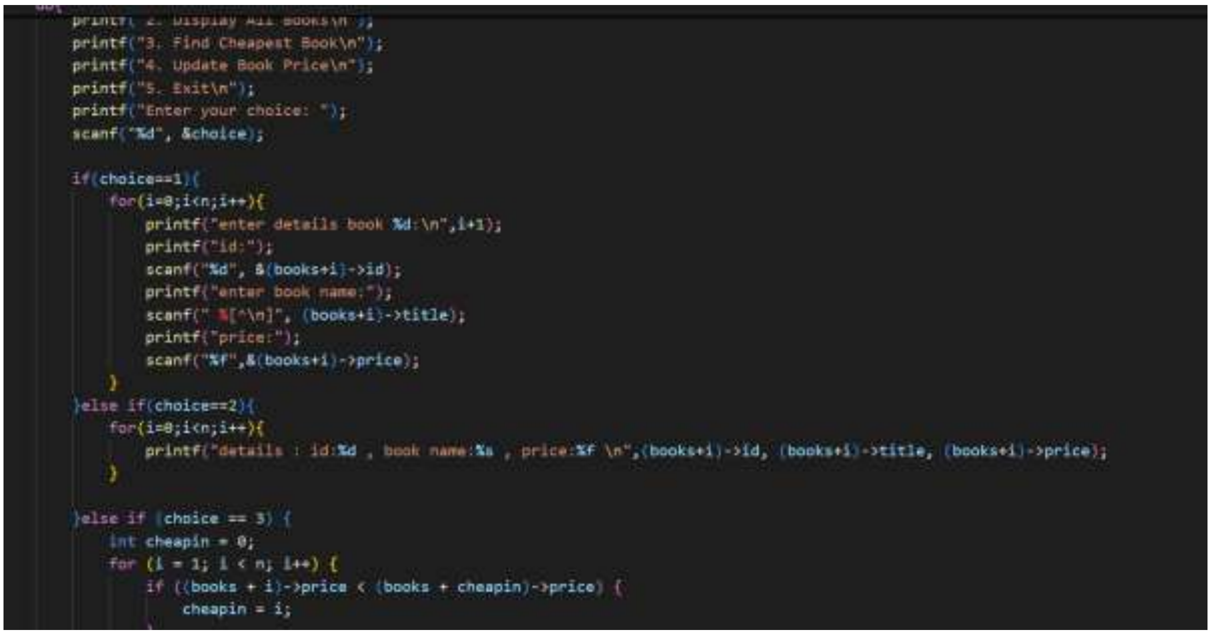
Display Details: Display the details of all books.

Find Cheapest Book: Identify and display the details of the cheapest book.

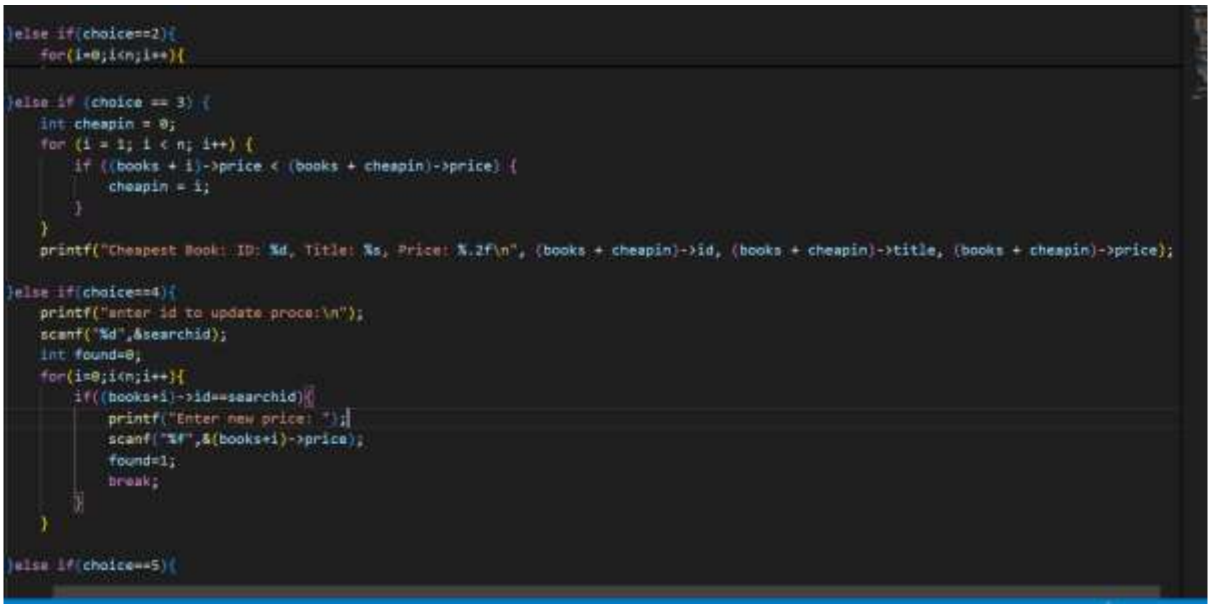
Update Price: Allow the user to update the price of a specific book by entering its ID



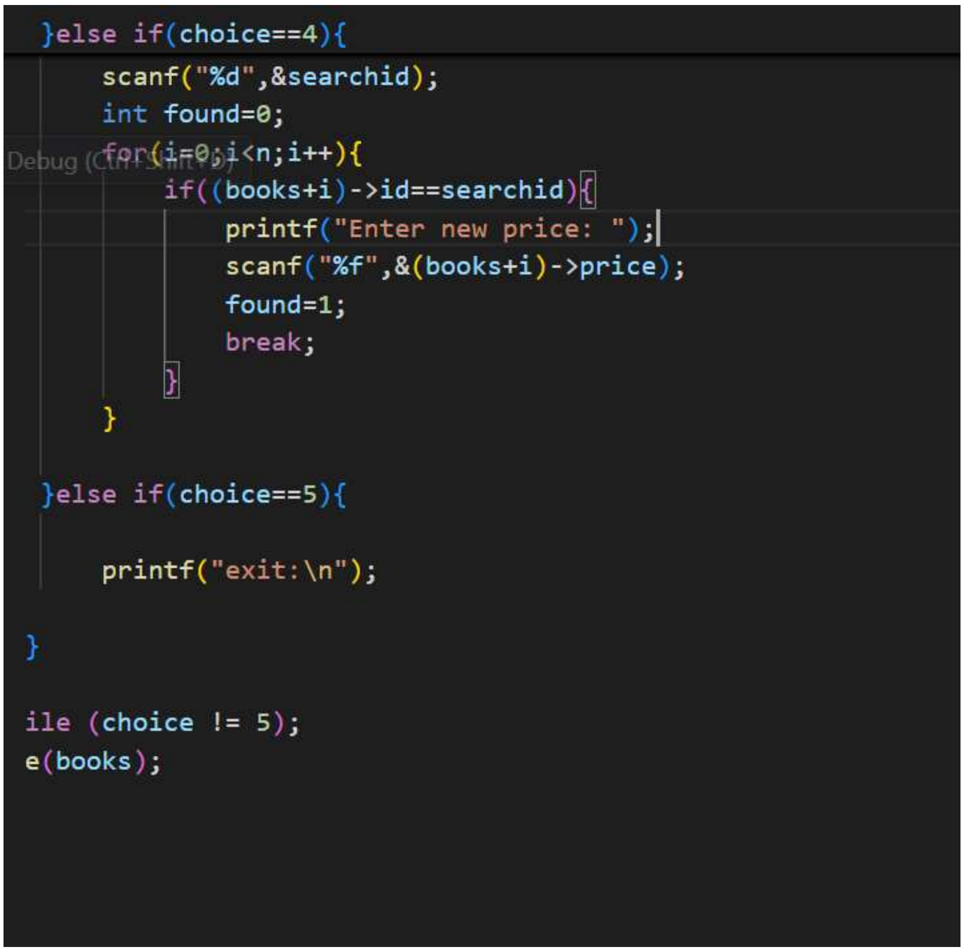
```c
#include<stdio.h>
#include<stdlib.h>

struct book{
    int id;
    char title[100];
    float price;

};

int main(){
    struct book *books;
    int i,choice,searchid,n;

    printf("enter num of books:");
    scanf("%d",&n);

    books=(struct book*)malloc(n * sizeof(struct book));

    if (books == NULL) {
        printf("Memory allocation failed!\n");
        return 1;
    }

    do{
        printf("menu:\n");
        printf("\nMenu:\n");
        printf("1. Input Book Details\n");
        printf("2. Display All Books\n");
```



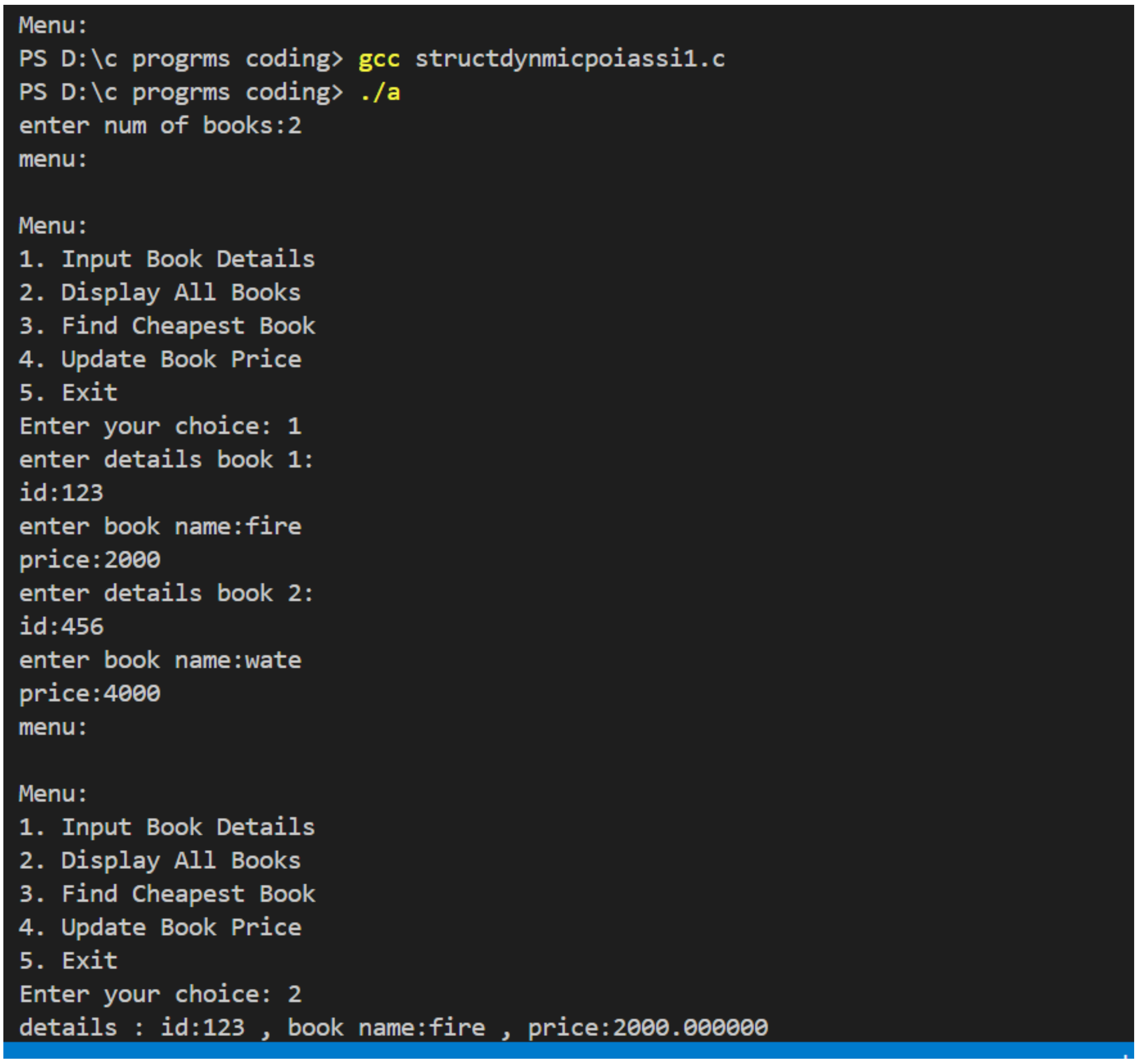
```c
        printf("3. Find Cheapest Book\n");
        printf("4. Update Book Price\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        if(choice==1){
            for(i=0;i<n;i++){
                printf("enter details book %d:\n",i+1);
                printf("id:");
                scanf("%d", &(books+i)->id);
                printf("enter book name:");
                scanf(" %[^\n]", (books+i)->title);
                printf("price:");
                scanf("%f",&(books+i)->price);
            }
        }else if(choice==2){
            for(i=0;i<n;i++){
                printf("details : id:%d , book name:%s , price:%f \n",(books+i)->id, (books+i)->title, (books+i)->price);
            }

        }else if (choice == 3) {
            int cheapin = 0;
            for (i = 1; i < n; i++) {
                if ((books + i)->price < (books + cheapin)->price) {
                    cheapin = i;
```



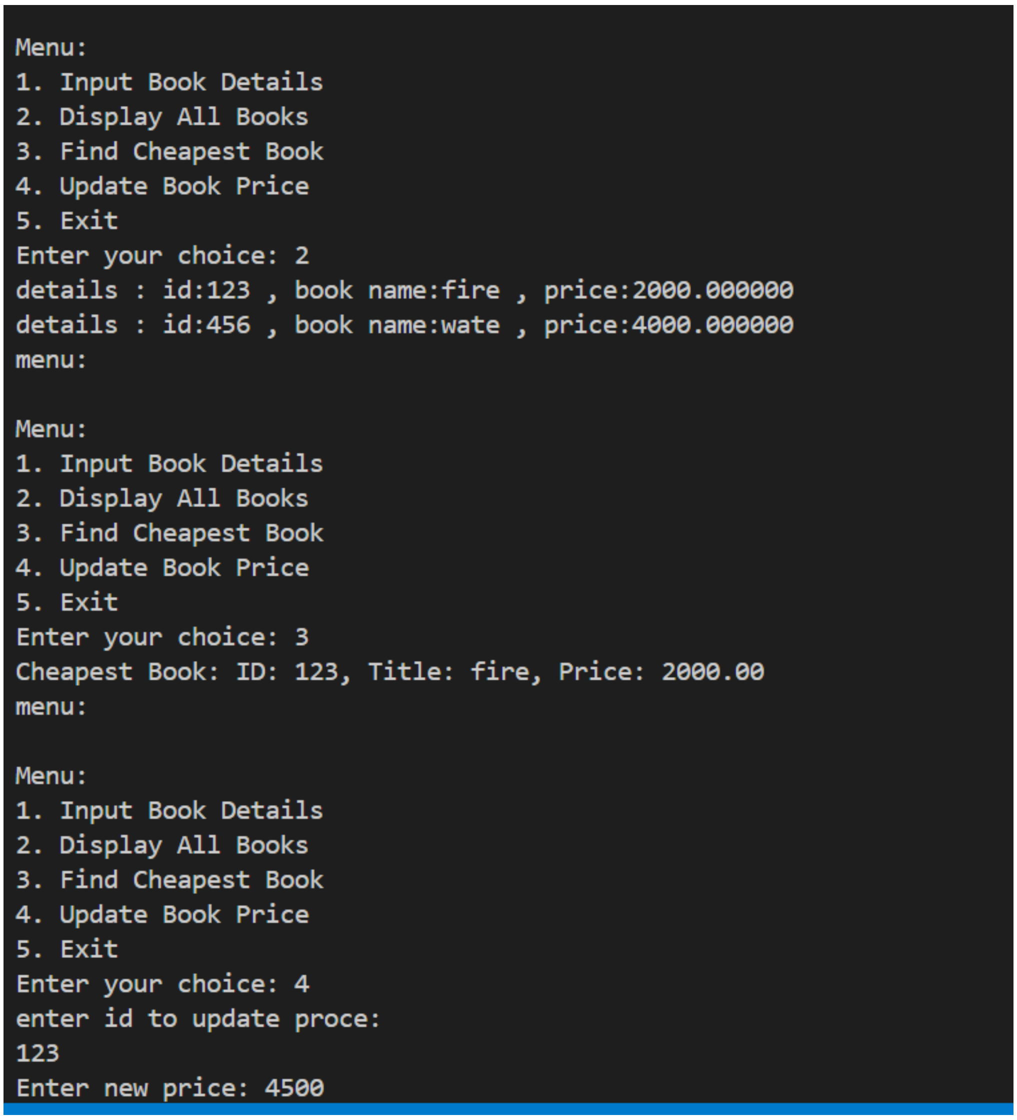
```c
}else if(choice==2){
    for(i=0;i<n;i++){

}else if (choice == 3) {
    int cheapin = 0;
    for (i = 1; i < n; i++) {
        if ((books + i)->price < (books + cheapin)->price) {
            cheapin = i;
        }
    }
    printf("Cheapest Book: ID: %d, Title: %s, Price: %.2f\n", (books + cheapin)->id, (books + cheapin)->title, (books + cheapin)->price);

}else if(choice==4){
    printf("enter id to update proce:\n");
    scanf("%d",&searchid);
    int found=0;
    for(i=0;i<n;i++){
        if((books+i)->id==searchid){
            printf("Enter new price: ");
            scanf("%f",&(books+i)->price);
            found=1;
            break;
        }
    }

}else if(choice==5){
```



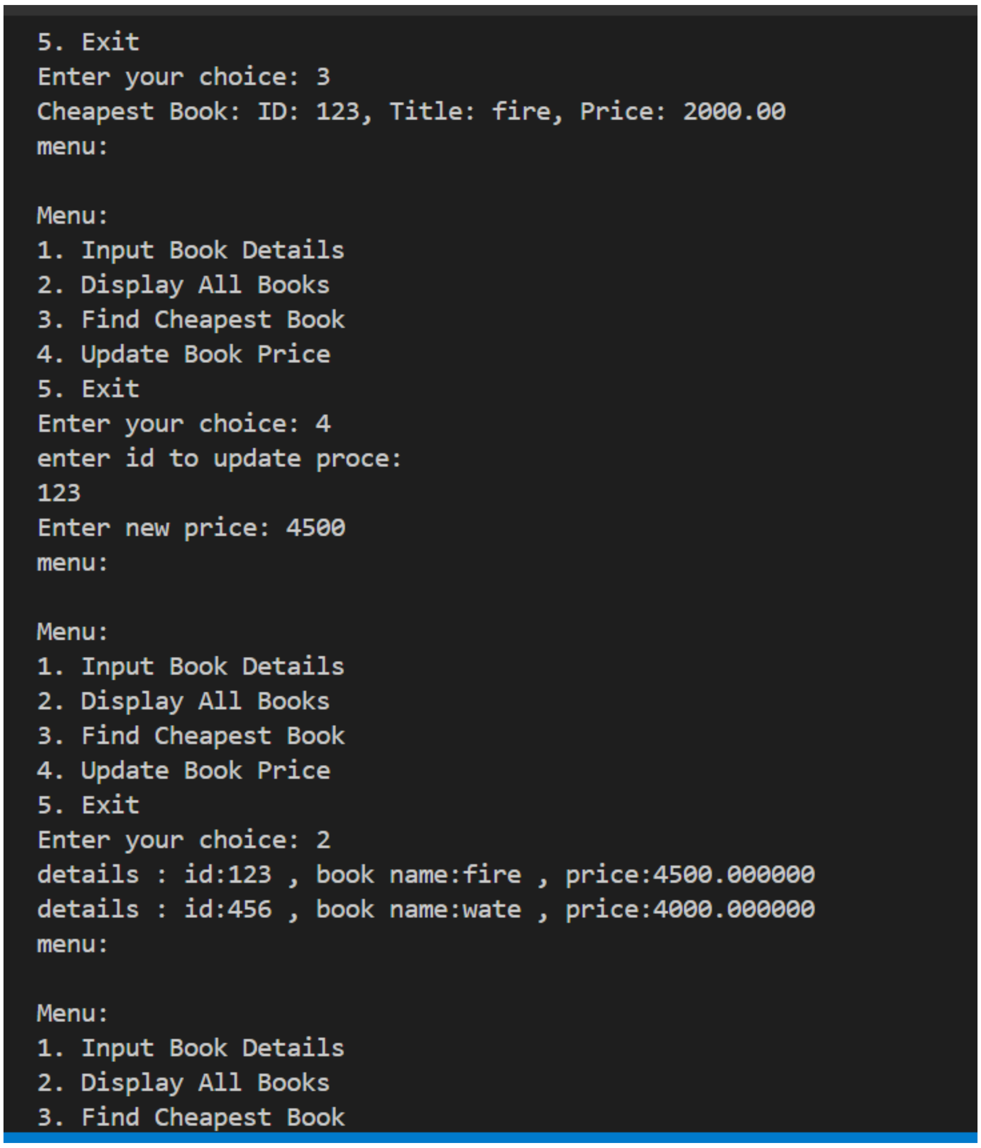
```c
 }else if(choice==4){
     scanf("%d",&searchid);
     int found=0;
     for(i=0;i<n;i++){
         if((books+i)->id==searchid){
             printf("Enter new price: ");
             scanf("%f",&(books+i)->price);
             found=1;
             break;
         }
     }

 }else if(choice==5){

     printf("exit:\n");

}

ile (choice != 5);
e(books);
```



```
Menu:
PS D:\c progrms coding> gcc structdynmicpoiassi1.c
PS D:\c progrms coding> ./a
enter num of books:2
menu:

Menu:
1. Input Book Details
2. Display All Books
3. Find Cheapest Book
4. Update Book Price
5. Exit
Enter your choice: 1
enter details book 1:
id:123
enter book name:fire
price:2000
enter details book 2:
id:456
enter book name:wate
price:4000
menu:

Menu:
1. Input Book Details
2. Display All Books
3. Find Cheapest Book
4. Update Book Price
5. Exit
Enter your choice: 2
details : id:123 , book name:fire , price:2000.000000
```



```
Menu:
1. Input Book Details
2. Display All Books
3. Find Cheapest Book
4. Update Book Price
5. Exit
Enter your choice: 2
details : id:123 , book name:fire , price:2000.000000
details : id:456 , book name:wate , price:4000.000000
menu:

Menu:
1. Input Book Details
2. Display All Books
3. Find Cheapest Book
4. Update Book Price
5. Exit
Enter your choice: 3
Cheapest Book: ID: 123, Title: fire, Price: 2000.00
menu:

Menu:
1. Input Book Details
2. Display All Books
3. Find Cheapest Book
4. Update Book Price
5. Exit
Enter your choice: 4
enter id to update proce:
123
Enter new price: 4500
```



```
5. Exit
Enter your choice: 3
Cheapest Book: ID: 123, Title: fire, Price: 2000.00
menu:

Menu:
1. Input Book Details
2. Display All Books
3. Find Cheapest Book
4. Update Book Price
5. Exit
Enter your choice: 4
enter id to update proce:
123
Enter new price: 4500
menu:

Menu:
1. Input Book Details
2. Display All Books
3. Find Cheapest Book
4. Update Book Price
5. Exit
Enter your choice: 2
details : id:123 , book name:fire , price:4500.000000
details : id:456 , book name:wate , price:4000.000000
menu:

Menu:
1. Input Book Details
2. Display All Books
3. Find Cheapest Book
```

2. Problem 2: Dynamic Point Array

Problem Statement:

Write a C program to handle a dynamic array of points in a 2D space using dynamic memory allocation. The program should:

Define a structure named Point with the following fields:

x (float): The x-coordinate of the point.

y (float): The y-coordinate of the point.

Dynamically allocate memory for n points (where n is input by the user).

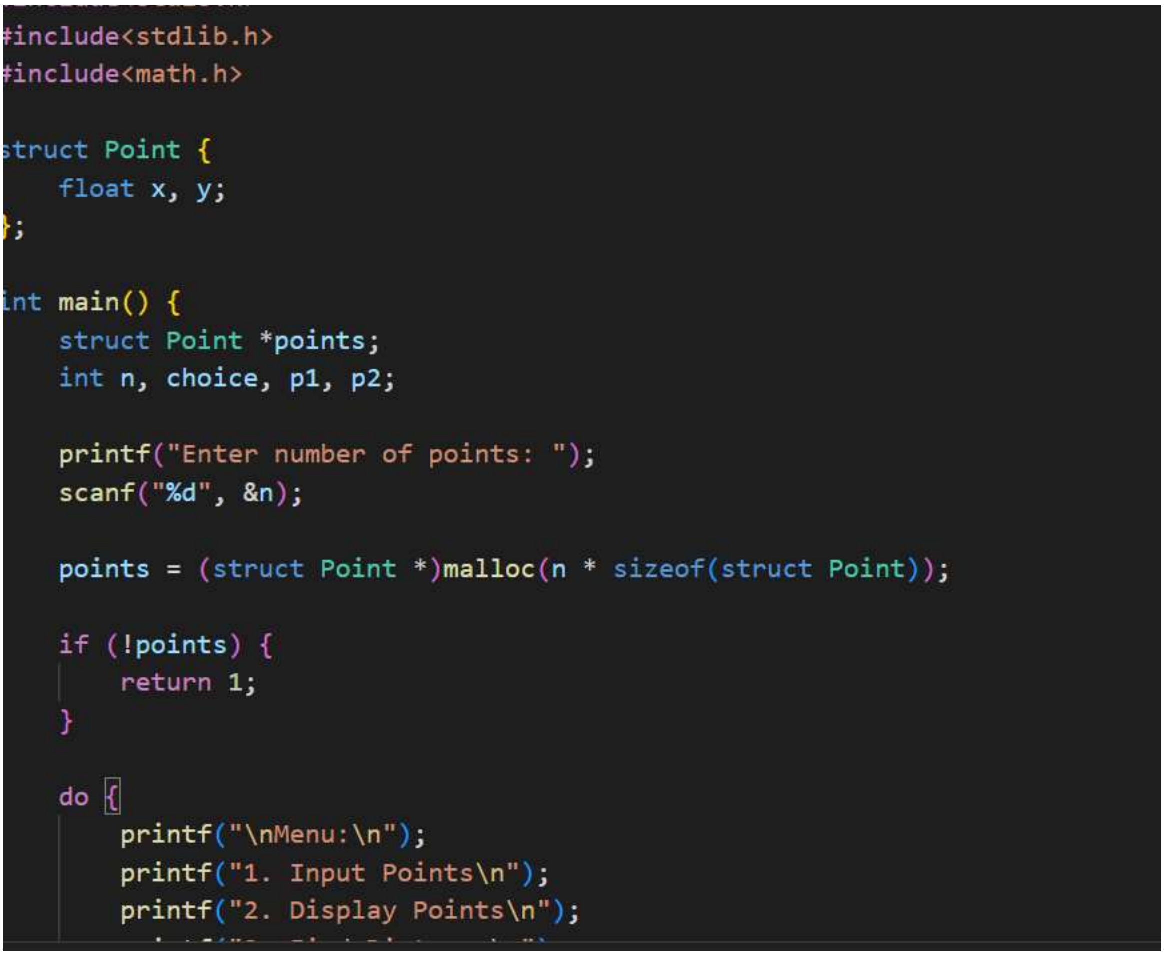
Implement the following features:

Input Details: Input the coordinates of each point.

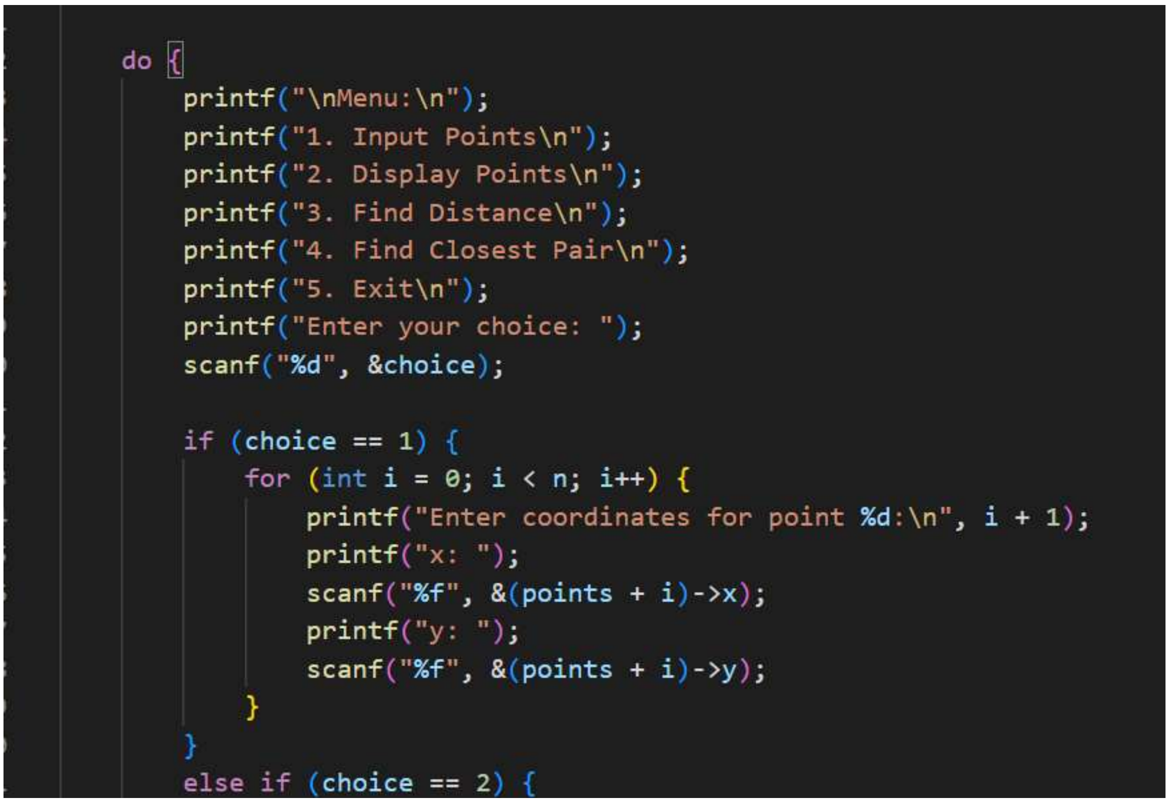
Display Points: Display the coordinates of all points.

Find Distance: Calculate the Euclidean distance between two points chosen by the user (by their indices in the array).

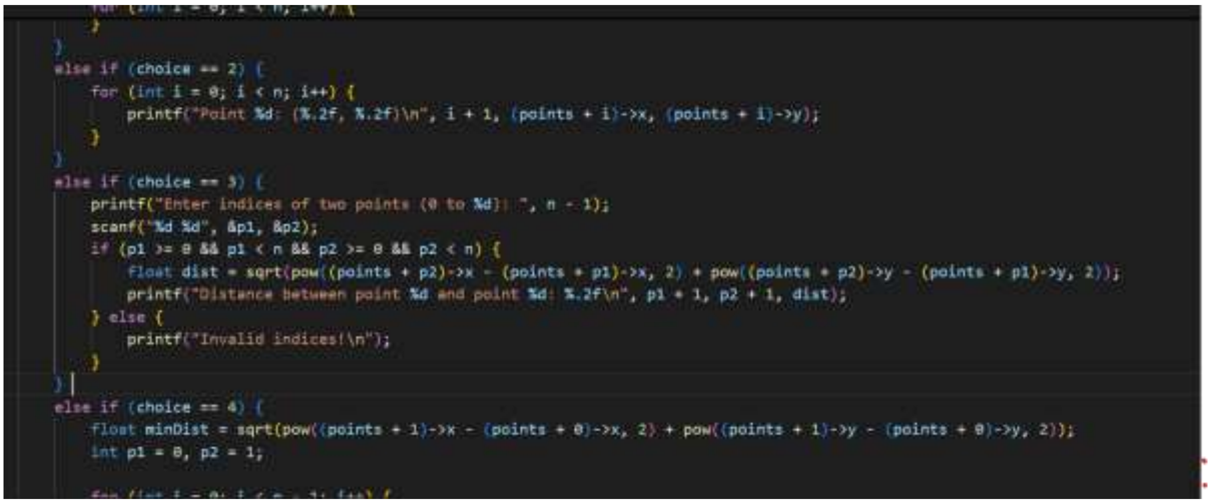
Find Closest Pair: Identify and display the pair of points that are closest to each other.



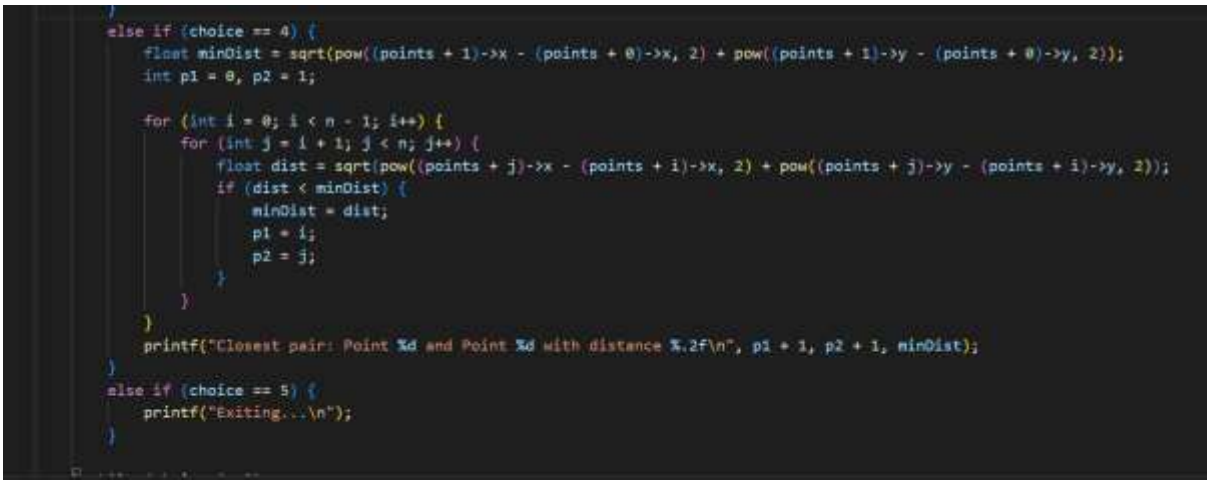
```c
#include<stdlib.h>
#include<math.h>

struct Point {
    float x, y;
};

int main() {
    struct Point *points;
    int n, choice, p1, p2;

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

    points = (struct Point *)malloc(n * sizeof(struct Point));

    if (!points) {
        return 1;
    }

    do {
        printf("\nMenu:\n");
        printf("1. Input Points\n");
        printf("2. Display Points\n");
```



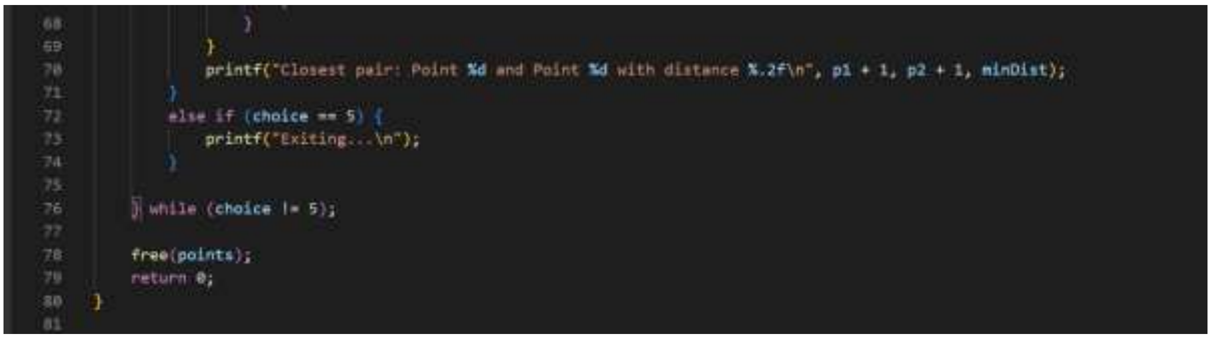
```c
do {
    printf("\nMenu:\n");
    printf("1. Input Points\n");
    printf("2. Display Points\n");
    printf("3. Find Distance\n");
    printf("4. Find Closest Pair\n");
    printf("5. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    if (choice == 1) {
        for (int i = 0; i < n; i++) {
            printf("Enter coordinates for point %d:\n", i + 1);
            printf("x: ");
            scanf("%f", &(points + i)->x);
            printf("y: ");
            scanf("%f", &(points + i)->y);
        }
    }
    else if (choice == 2) {
```



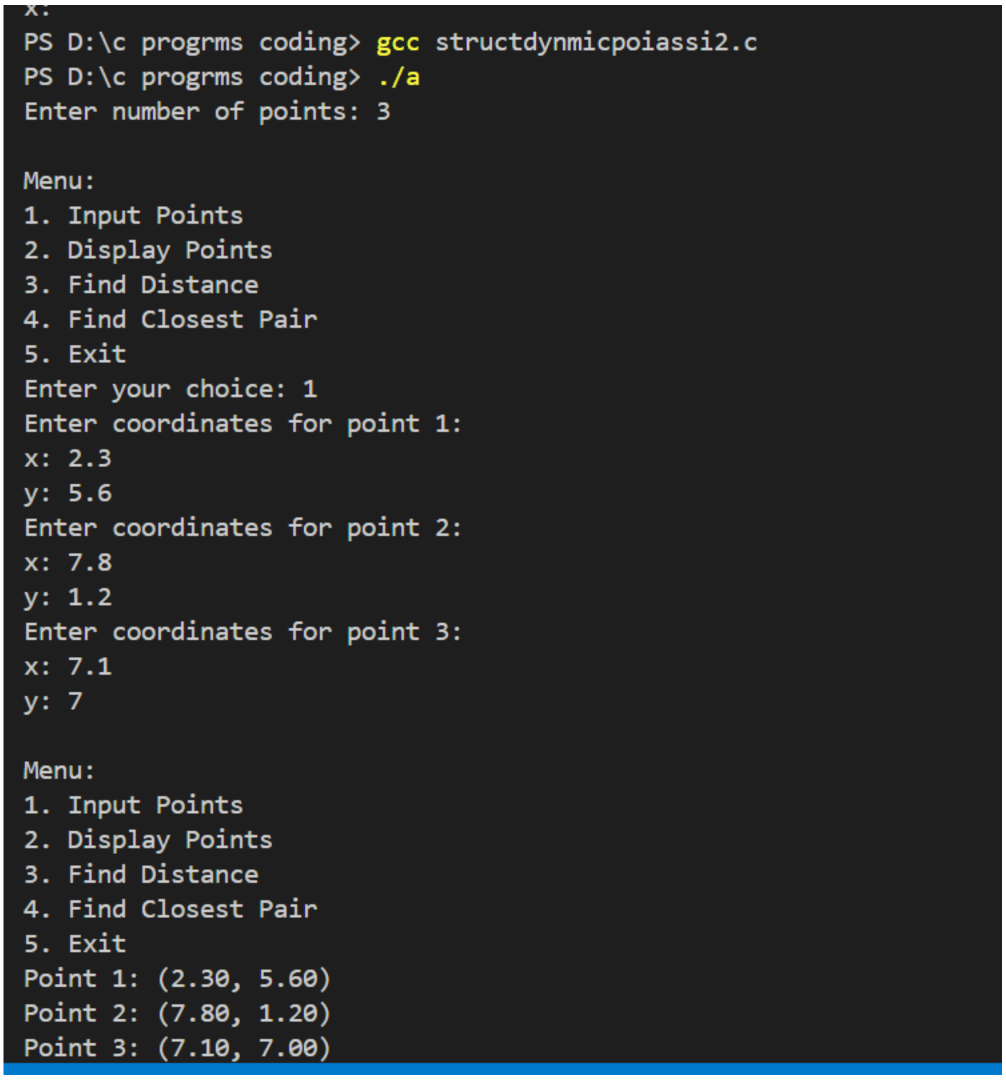
```c
        }
    }
    else if (choice == 2) {
        for (int i = 0; i < n; i++) {
            printf("Point %d: (%.2f, %.2f)\n", i + 1, (points + i)->x, (points + i)->y);
        }
    }
    else if (choice == 3) {
        printf("Enter indices of two points (0 to %d): ", n - 1);
        scanf("%d %d", &p1, &p2);
        if (p1 >= 0 && p1 < n && p2 >= 0 && p2 < n) {
            float dist = sqrt(pow((points + p2)->x - (points + p1)->x, 2) + pow((points + p2)->y - (points + p1)->y, 2));
            printf("Distance between point %d and point %d: %.2f\n", p1 + 1, p2 + 1, dist);
        } else {
            printf("Invalid indices!\n");
        }
    }
    else if (choice == 4) {
        float minDist = sqrt(pow((points + 1)->x - (points + 0)->x, 2) + pow((points + 1)->y - (points + 0)->y, 2));
        int p1 = 0, p2 = 1;
```



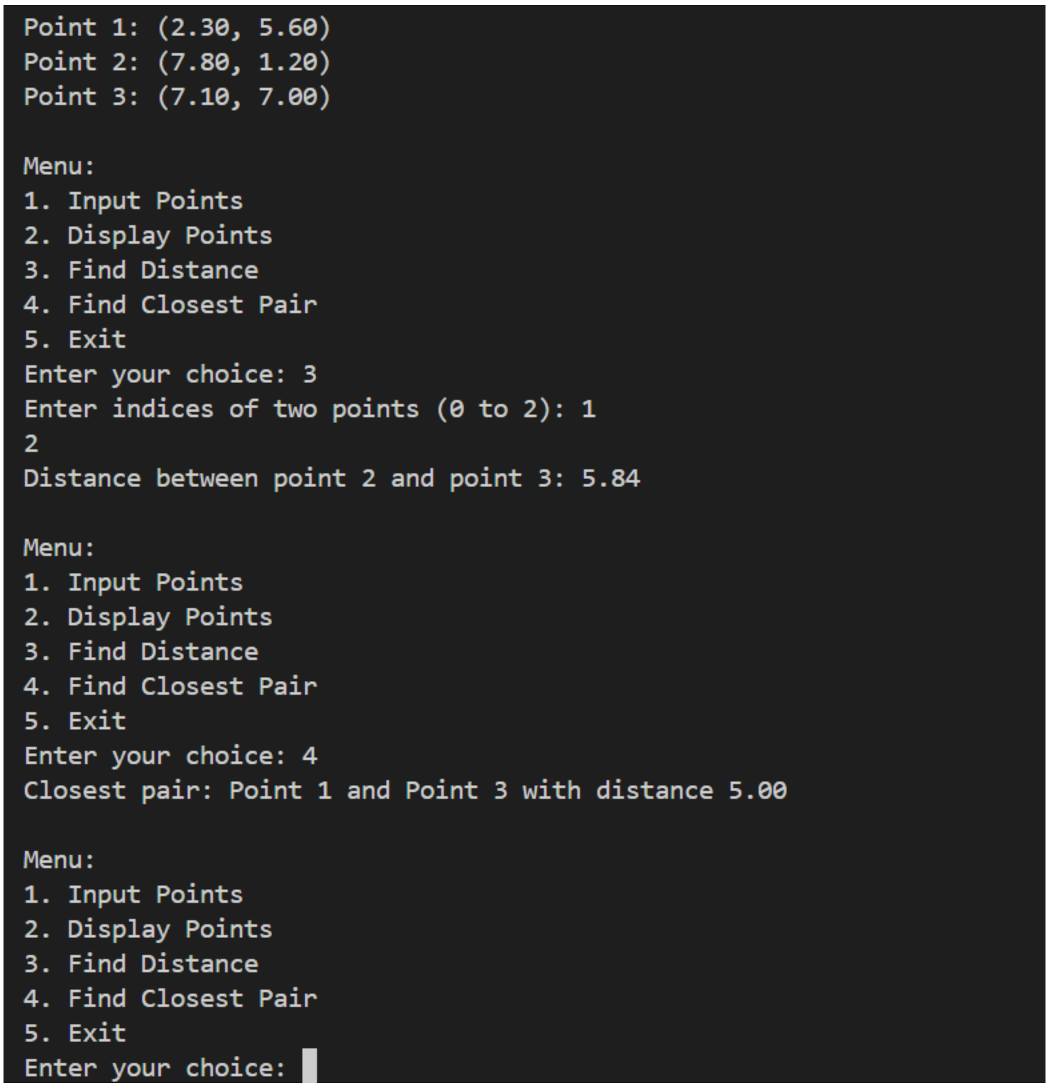
```c
    }
    else if (choice == 4) {
        float minDist = sqrt(pow((points + 1)->x - (points + 0)->x, 2) + pow((points + 1)->y - (points + 0)->y, 2));
        int p1 = 0, p2 = 1;

        for (int i = 0; i < n - 1; i++) {
            for (int j = i + 1; j < n; j++) {
                float dist = sqrt(pow((points + j)->x - (points + i)->x, 2) + pow((points + j)->y - (points + i)->y, 2));
                if (dist < minDist) {
                    minDist = dist;
                    p1 = i;
                    p2 = j;
                }
            }
        }
        printf("Closest pair: Point %d and Point %d with distance %.2f\n", p1 + 1, p2 + 1, minDist);
    }
    else if (choice == 5) {
        printf("Exiting...\n");
    }
```



```c
            }
        }
        printf("Closest pair: Point %d and Point %d with distance %.2f\n", p1 + 1, p2 + 1, minDist);
    }
    else if (choice == 5) {
        printf("Exiting...\n");
    }

} while (choice != 5);

free(points);
return 0;
}
```



```
x:
PS D:\c progrms coding> gcc structdynmicpoiassi2.c
PS D:\c progrms coding> ./a
Enter number of points: 3

Menu:
1. Input Points
2. Display Points
3. Find Distance
4. Find Closest Pair
5. Exit
Enter your choice: 1
Enter coordinates for point 1:
x: 2.3
y: 5.6
Enter coordinates for point 2:
x: 7.8
y: 1.2
Enter coordinates for point 3:
x: 7.1
y: 7

Menu:
1. Input Points
2. Display Points
3. Find Distance
4. Find Closest Pair
5. Exit
Point 1: (2.30, 5.60)
Point 2: (7.80, 1.20)
Point 3: (7.10, 7.00)
```



```
Point 1: (2.30, 5.60)
Point 2: (7.80, 1.20)
Point 3: (7.10, 7.00)

Menu:
1. Input Points
2. Display Points
3. Find Distance
4. Find Closest Pair
5. Exit
Enter your choice: 3
Enter indices of two points (0 to 2): 1
2
Distance between point 2 and point 3: 5.84

Menu:
1. Input Points
2. Display Points
3. Find Distance
4. Find Closest Pair
5. Exit
Enter your choice: 4
Closest pair: Point 1 and Point 3 with distance 5.00

Menu:
1. Input Points
2. Display Points
3. Find Distance
4. Find Closest Pair
5. Exit
Enter your choice: 
```

3. Problem Statement: Vehicle Registration System

Write a C program to simulate a vehicle registration system using unions to handle different types of vehicles. The program should:

Define a union named Vehicle with the following members:

car_model (character array of size 50): To store the model name of a car.

bike_cc (integer): To store the engine capacity (in CC) of a bike.

bus_seats (integer): To store the number of seats in a bus.

Create a structure VehicleInfo that contains:

type (character): To indicate the type of vehicle (C for car, B for bike, S for bus).

Vehicle (the union defined above): To store the specific details of the vehicle based on its type.

Implement the following features:

Input Details: Prompt the user to input the type of vehicle and its corresponding details:

For a car: Input the model name.

For a bike: Input the engine capacity.

For a bus: Input the number of seats.

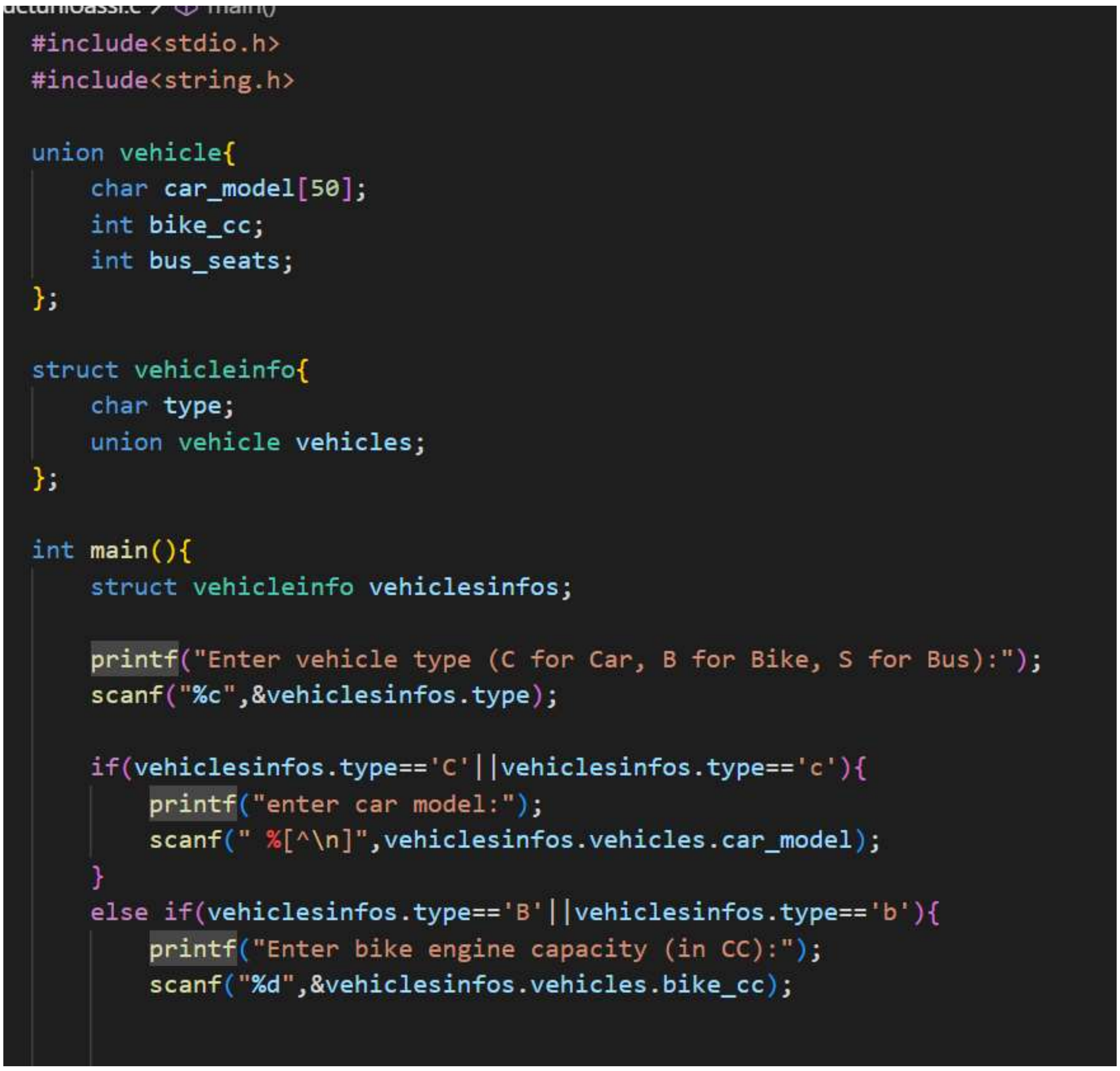
Display Details: Display the details of the vehicle based on its type.

Use the union effectively to save memory and ensure only relevant information is stored.

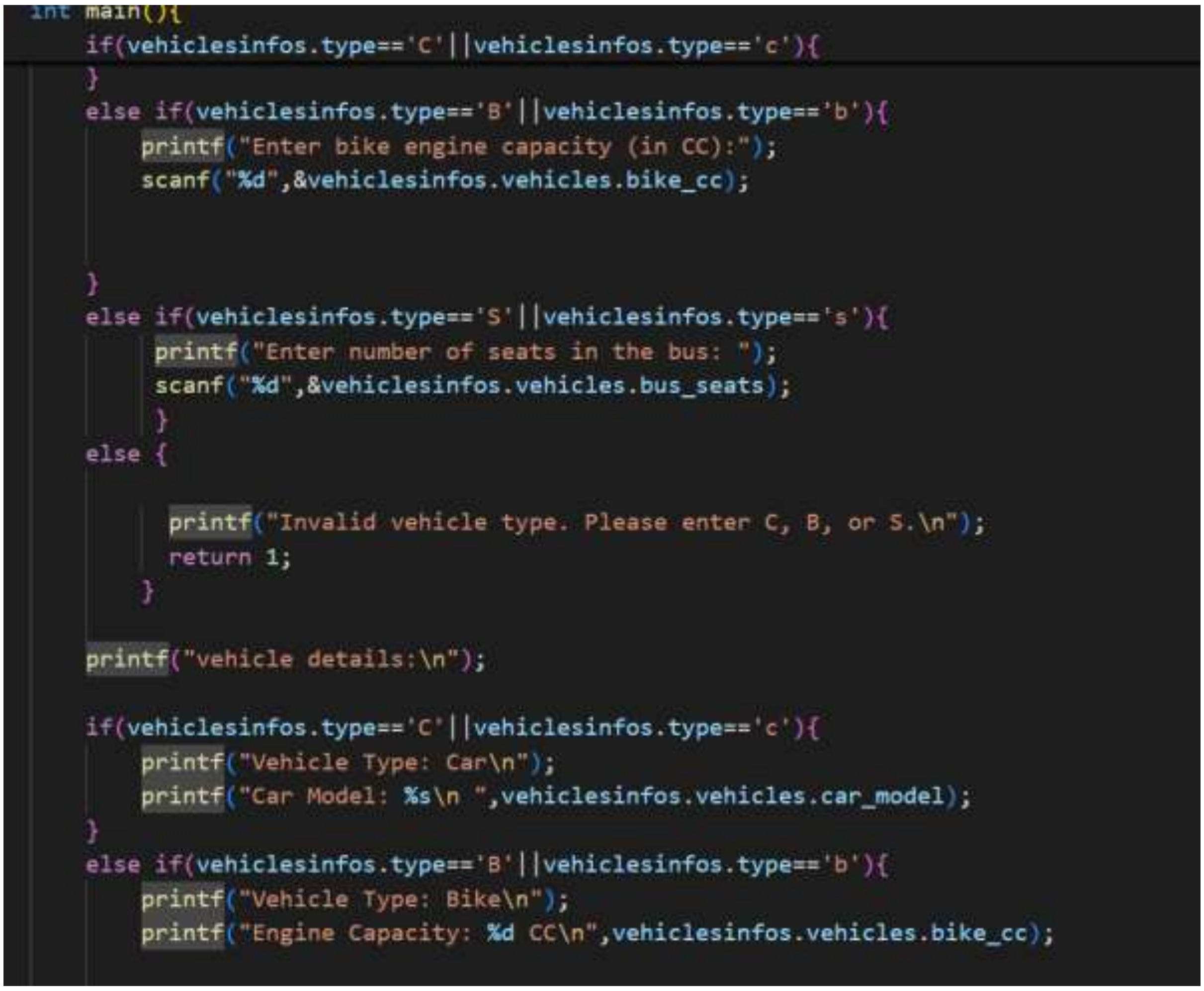
Constraints

The type of vehicle should be one of C, B, or S.

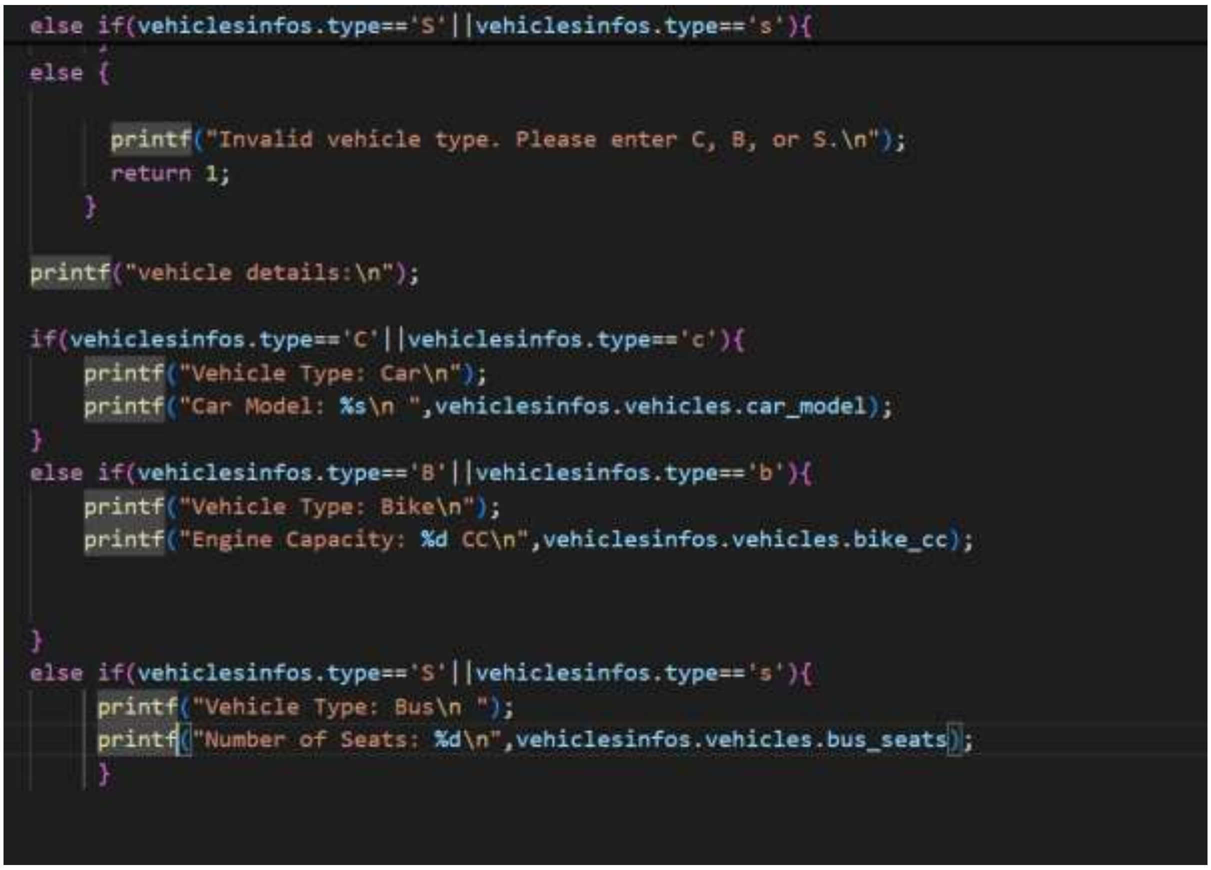
For invalid input, prompt the user again.



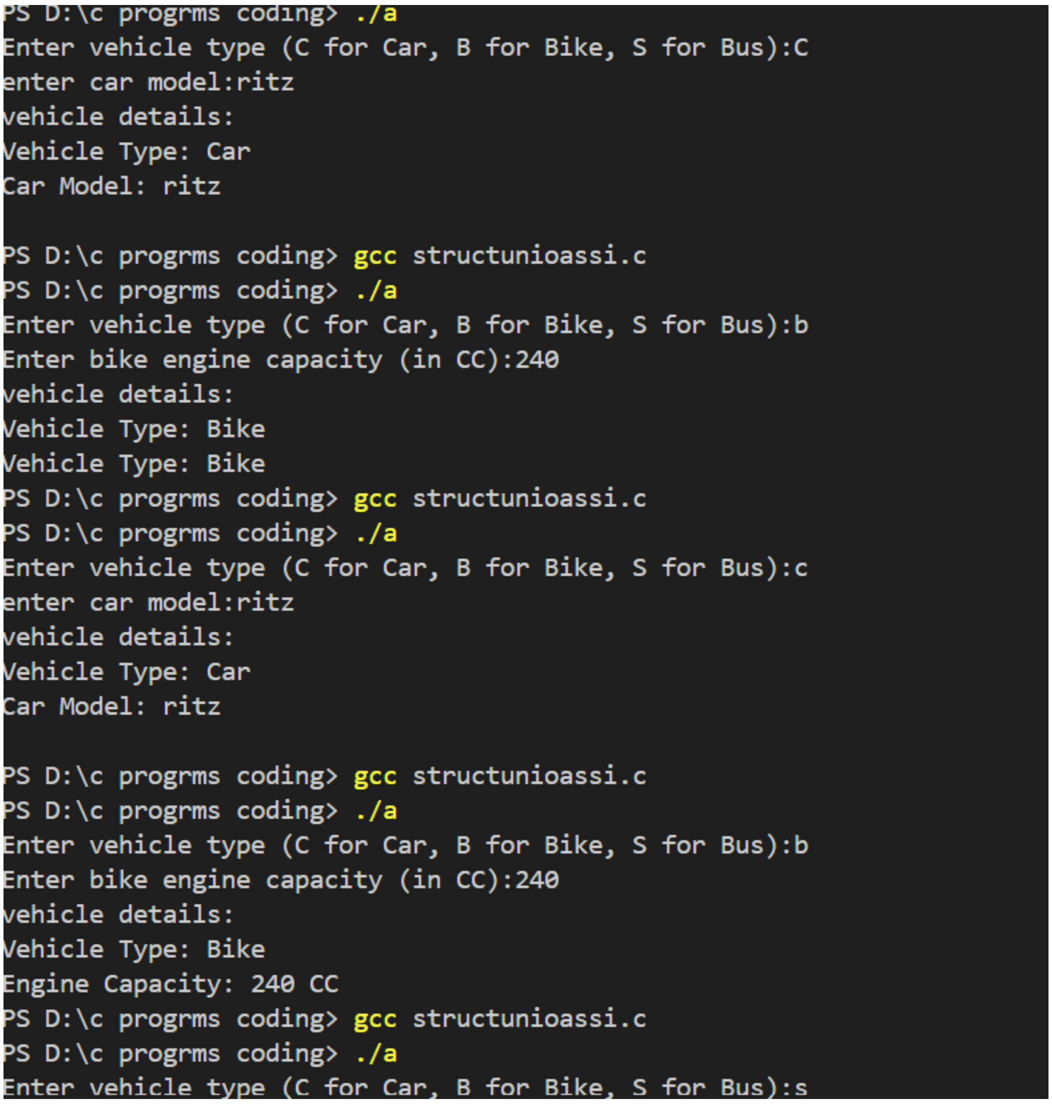
```c
#include<stdio.h>
#include<string.h>

union vehicle{
    char car_model[50];
    int bike_cc;
    int bus_seats;
};

struct vehicleinfo{
    char type;
    union vehicle vehicles;
};

int main(){
    struct vehicleinfo vehiclesinfos;

    printf("Enter vehicle type (C for Car, B for Bike, S for Bus):");
    scanf("%c",&vehiclesinfos.type);

    if(vehiclesinfos.type=='C'||vehiclesinfos.type=='c'){
        printf("enter car model:");
        scanf(" %[^\n]",vehiclesinfos.vehicles.car_model);
    }
    else if(vehiclesinfos.type=='B'||vehiclesinfos.type=='b'){
        printf("Enter bike engine capacity (in CC):");
        scanf("%d",&vehiclesinfos.vehicles.bike_cc);
```



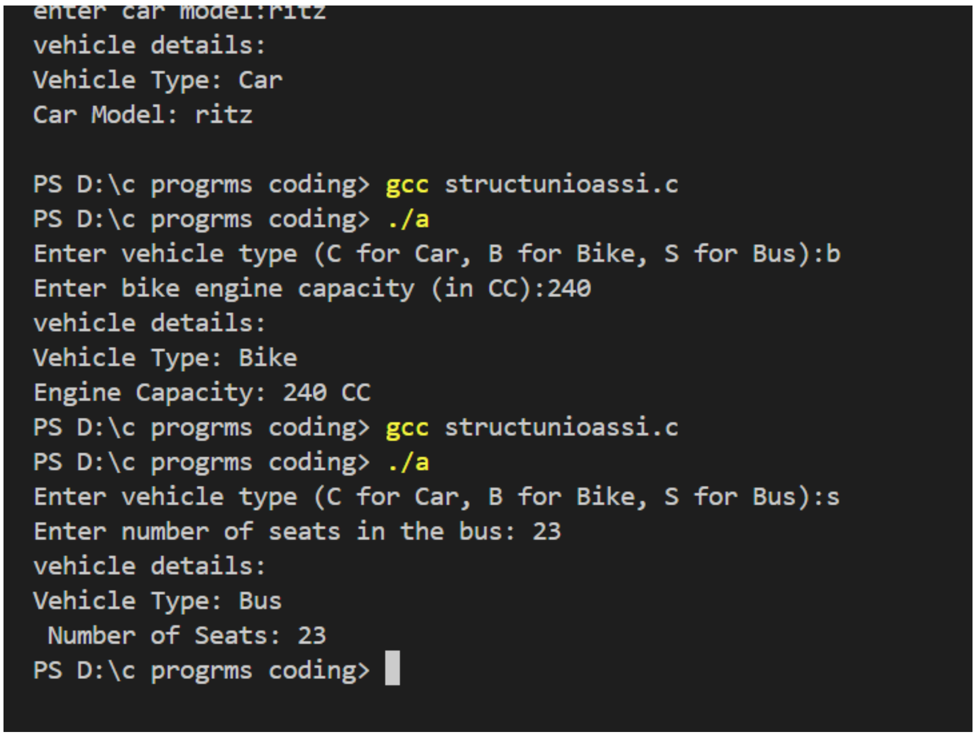
```
int main(){
    if(vehiclesinfos.type=='C'||vehiclesinfos.type=='c'){
    }
    else if(vehiclesinfos.type=='B'||vehiclesinfos.type=='b'){
        printf("Enter bike engine capacity (in CC):");
        scanf("%d",&vehiclesinfos.vehicles.bike_cc);


    }
    else if(vehiclesinfos.type=='S'||vehiclesinfos.type=='s'){
         printf("Enter number of seats in the bus: ");
         scanf("%d",&vehiclesinfos.vehicles.bus_seats);
         }
    else {

          printf("Invalid vehicle type. Please enter C, B, or S.\n");
          return 1;
        }

    printf("vehicle details:\n");

    if(vehiclesinfos.type=='C'||vehiclesinfos.type=='c'){
        printf("Vehicle Type: Car\n");
        printf("Car Model: %s\n ",vehiclesinfos.vehicles.car_model);
    }
    else if(vehiclesinfos.type=='B'||vehiclesinfos.type=='b'){
        printf("Vehicle Type: Bike\n");
        printf("Engine Capacity: %d CC\n",vehiclesinfos.vehicles.bike_cc);
```



```
else if(vehiclesinfos.type=='S'||vehiclesinfos.type=='s'){
else {

        printf("Invalid vehicle type. Please enter C, B, or S.\n");
        return 1;
    }

printf("vehicle details:\n");

if(vehiclesinfos.type=='C'||vehiclesinfos.type=='c'){
    printf("Vehicle Type: Car\n");
    printf("Car Model: %s\n ",vehiclesinfos.vehicles.car_model);
}
else if(vehiclesinfos.type=='B'||vehiclesinfos.type=='b'){
    printf("Vehicle Type: Bike\n");
    printf("Engine Capacity: %d CC\n",vehiclesinfos.vehicles.bike_cc);


}
else if(vehiclesinfos.type=='S'||vehiclesinfos.type=='s'){
     printf("Vehicle Type: Bus\n ");
     printf("Number of Seats: %d\n",vehiclesinfos.vehicles.bus_seats);
     }
```



```
PS D:\c progrms coding> ./a
Enter vehicle type (C for Car, B for Bike, S for Bus):C
enter car model:ritz
vehicle details:
Vehicle Type: Car
Car Model: ritz

PS D:\c progrms coding> gcc structunioassi.c
PS D:\c progrms coding> ./a
Enter vehicle type (C for Car, B for Bike, S for Bus):b
Enter bike engine capacity (in CC):240
vehicle details:
Vehicle Type: Bike
Vehicle Type: Bike
PS D:\c progrms coding> gcc structunioassi.c
PS D:\c progrms coding> ./a
Enter vehicle type (C for Car, B for Bike, S for Bus):c
enter car model:ritz
vehicle details:
Vehicle Type: Car
Car Model: ritz

PS D:\c progrms coding> gcc structunioassi.c
PS D:\c progrms coding> ./a
Enter vehicle type (C for Car, B for Bike, S for Bus):b
Enter bike engine capacity (in CC):240
vehicle details:
Vehicle Type: Bike
Engine Capacity: 240 CC
PS D:\c progrms coding> gcc structunioassi.c
PS D:\c progrms coding> ./a
Enter vehicle type (C for Car, B for Bike, S for Bus):s
```



```
enter car model:ritz
vehicle details:
Vehicle Type: Car
Car Model: ritz

PS D:\c progrms coding> gcc structunioassi.c
PS D:\c progrms coding> ./a
Enter vehicle type (C for Car, B for Bike, S for Bus):b
Enter bike engine capacity (in CC):240
vehicle details:
Vehicle Type: Bike
Engine Capacity: 240 CC
PS D:\c progrms coding> gcc structunioassi.c
PS D:\c progrms coding> ./a
Enter vehicle type (C for Car, B for Bike, S for Bus):s
Enter number of seats in the bus: 23
vehicle details:
Vehicle Type: Bus
 Number of Seats: 23
PS D:\c progrms coding> 
```

4. Problem Statement: Employee Records Management

Write a C program to manage a list of employees using dynamic memory allocation. The program should:

Define a structure named Employee with the following fields:

id (integer): A unique identifier for the employee.

name (character array of size 50): The employee's name.

salary (float): The employee's salary.

Dynamically allocate memory for storing information about n employees (where n is input by the user).

Implement the following features:

Input Details: Allow the user to input the details of each employee (ID, name, and salary).

Display Details: Display the details of all employees.

Search by ID: Allow the user to search for an employee by their ID and display their details.

Free Memory: Ensure that all dynamically allocated memory is freed at the end of the program.

Constraints

n (number of employees) must be a positive integer.

Employee IDs are unique.

Sample Input/Output

Input:

Enter the number of employees: 3

Enter details of employee 1:

ID: 101

Name: Alice

Salary: 50000

Enter details of employee 2:

ID: 102

Name: Bob

Salary: 60000

Enter details of employee 3:

ID: 103

Name: Charlie

Salary: 55000

Enter ID to search for: 102

Output:

Employee Details:

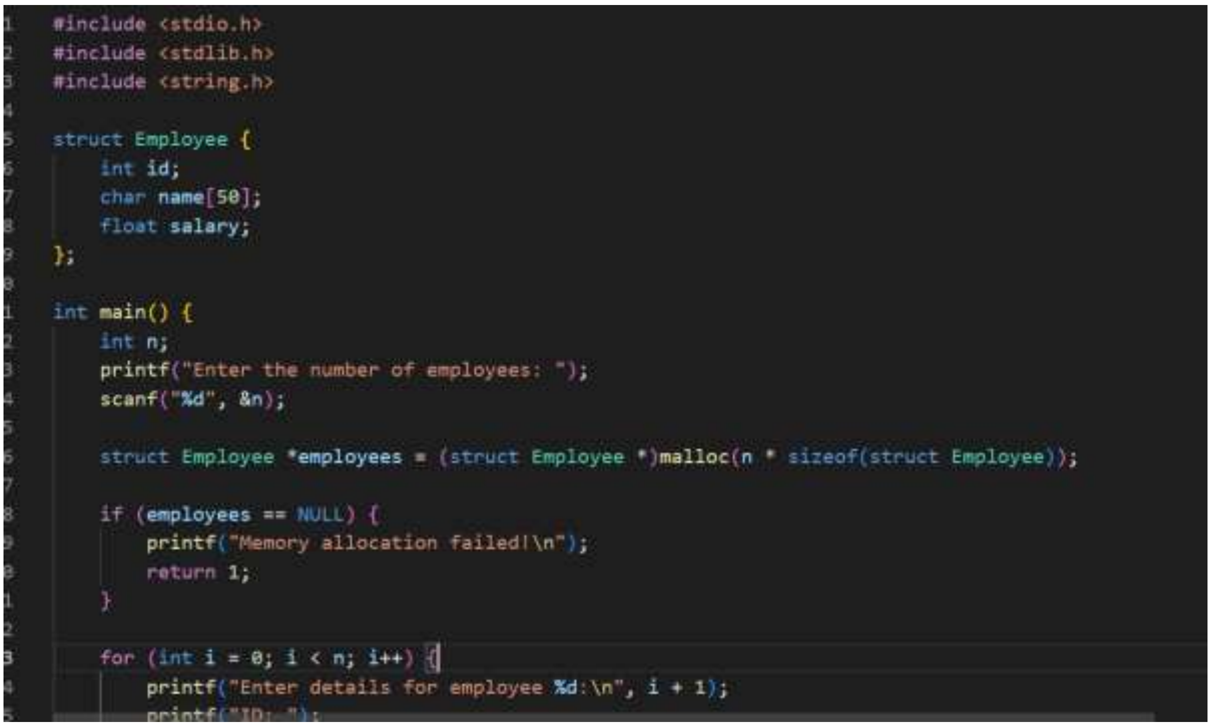
ID: 101, Name: Alice, Salary: 50000.00

ID: 102, Name: Bob, Salary: 60000.00

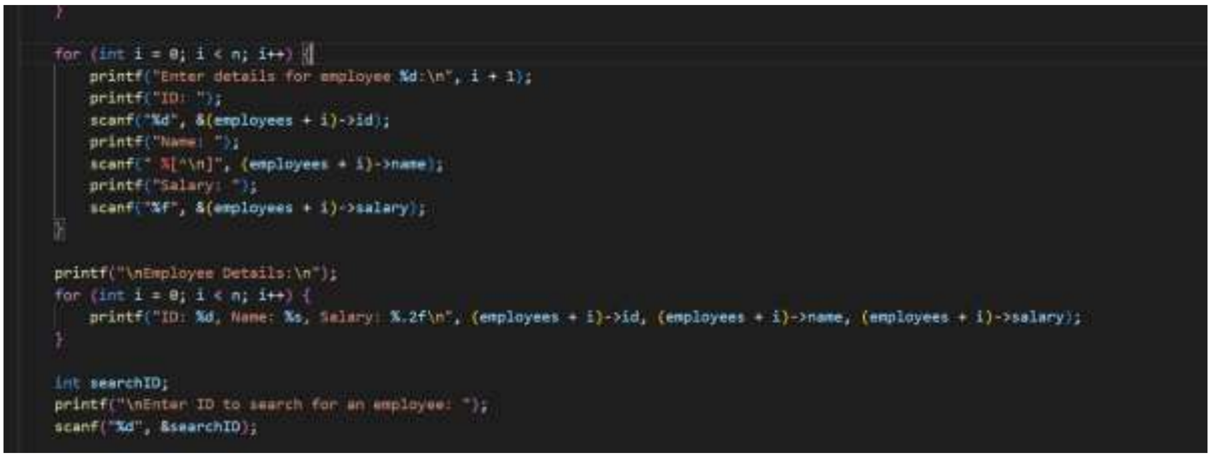
ID: 103, Name: Charlie, Salary: 55000.00

Search Result:

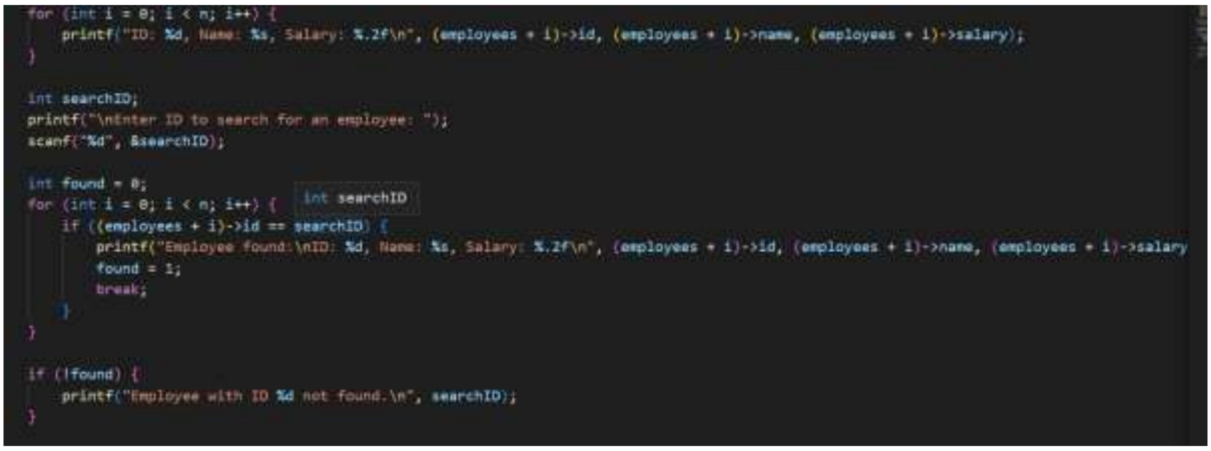
ID: 102, Name: Bob, Salary: 60000.00



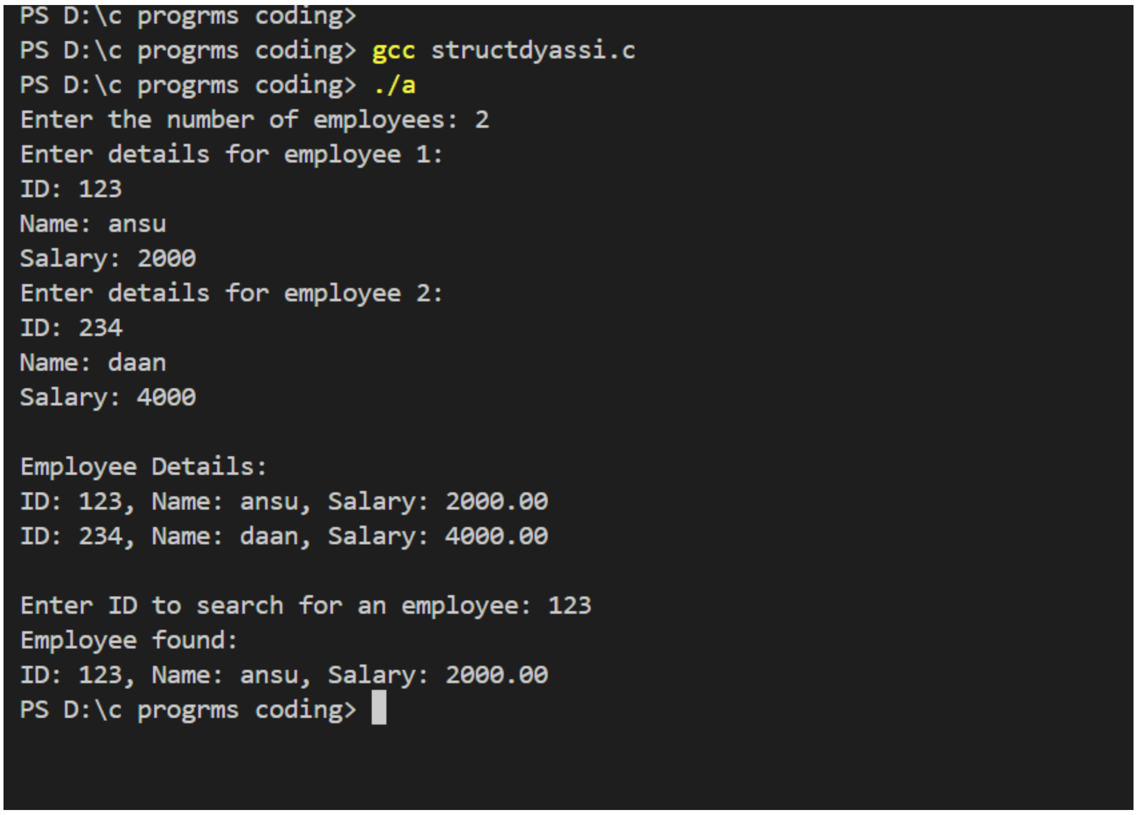
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

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

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

    struct Employee *employees = (struct Employee *)malloc(n * sizeof(struct Employee));

    if (employees == NULL) {
        printf("Memory allocation failed!\n");
        return 1;
    }

    for (int i = 0; i < n; i++) {
        printf("Enter details for employee %d:\n", i + 1);
        printf("ID: ");
```



```c
    }

    for (int i = 0; i < n; i++) {
        printf("Enter details for employee %d:\n", i + 1);
        printf("ID: ");
        scanf("%d", &(employees + i)->id);
        printf("Name: ");
        scanf(" %[^\n]", (employees + i)->name);
        printf("Salary: ");
        scanf("%f", &(employees + i)->salary);
    }

    printf("\nEmployee Details:\n");
    for (int i = 0; i < n; i++) {
        printf("ID: %d, Name: %s, Salary: %.2f\n", (employees + i)->id, (employees + i)->name, (employees + i)->salary);
    }

    int searchID;
    printf("\nEnter ID to search for an employee: ");
    scanf("%d", &searchID);
```



```c
for (int i = 0; i < n; i++) {
    printf("ID: %d, Name: %s, Salary: %.2f\n", (employees + i)->id, (employees + i)->name, (employees + i)->salary);
}

int searchID;
printf("\nEnter ID to search for an employee: ");
scanf("%d", &searchID);

int found = 0;
for (int i = 0; i < n; i++) {   int searchID
    if ((employees + i)->id == searchID) {
        printf("Employee found:\nID: %d, Name: %s, Salary: %.2f\n", (employees + i)->id, (employees + i)->name, (employees + i)->salary
        found = 1;
        break;
    }
}

if (!found) {
    printf("Employee with ID %d not found.\n", searchID);
}
```



```
PS D:\c progrms coding>
PS D:\c progrms coding> gcc structdyassi.c
PS D:\c progrms coding> ./a
Enter the number of employees: 2
Enter details for employee 1:
ID: 123
Name: ansu
Salary: 2000
Enter details for employee 2:
ID: 234
Name: daan
Salary: 4000

Employee Details:
ID: 123, Name: ansu, Salary: 2000.00
ID: 234, Name: daan, Salary: 4000.00

Enter ID to search for an employee: 123
Employee found:
ID: 123, Name: ansu, Salary: 2000.00
PS D:\c progrms coding>
```

5. Problem 1: Traffic Light System

Problem Statement:

Write a C program to simulate a traffic light system using enum. The program should:

Define an enum named TrafficLight with the values RED, YELLOW, and GREEN.

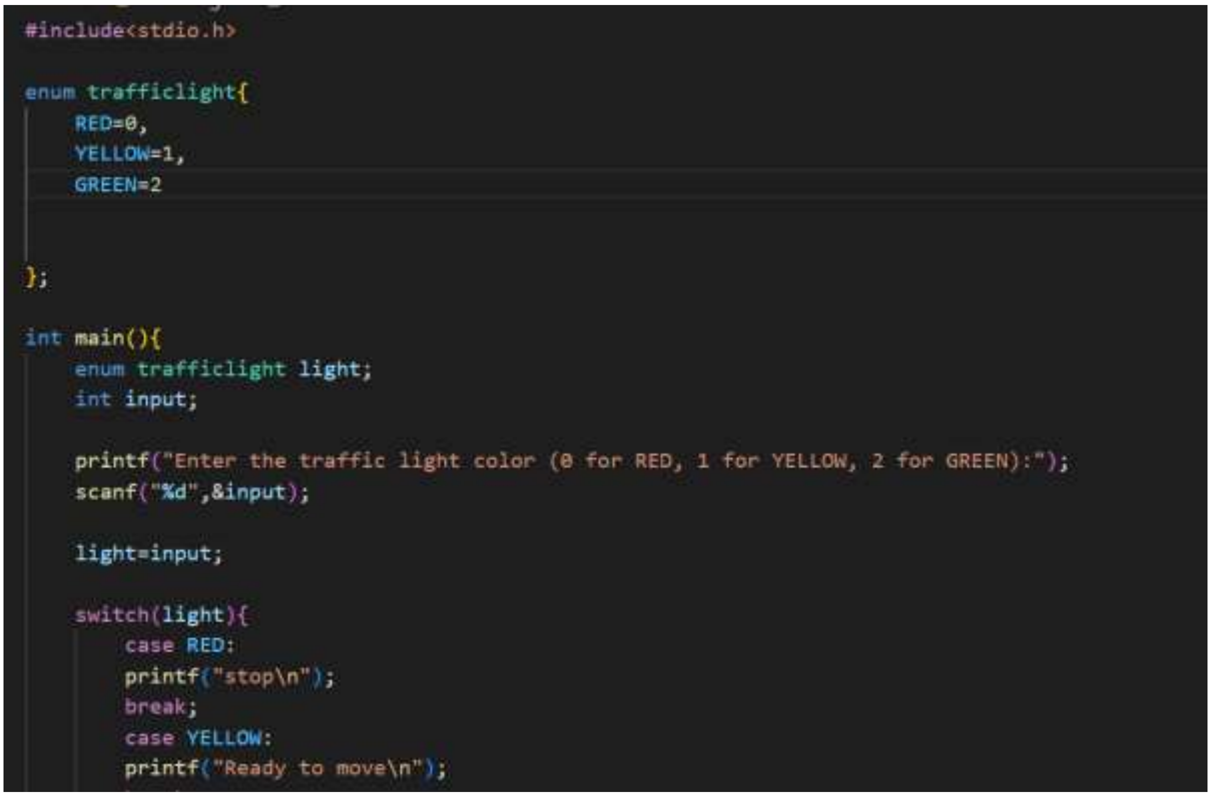
Accept the current light color as input from the user (as an integer: 0 for RED, 1 for YELLOW, 2 for GREEN).

Display an appropriate message based on the current light:

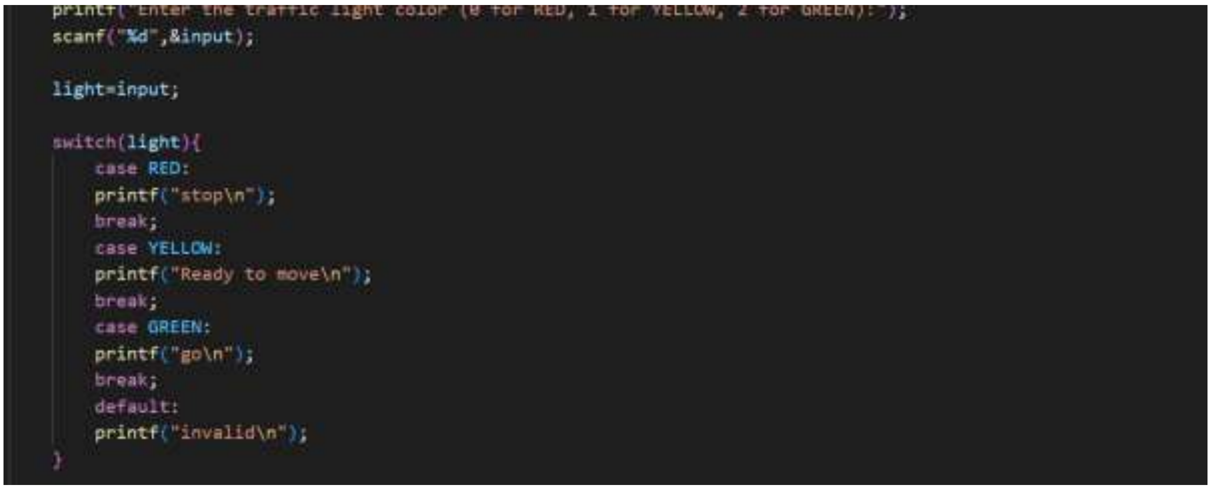
RED: "Stop"

YELLOW: "Ready to move"

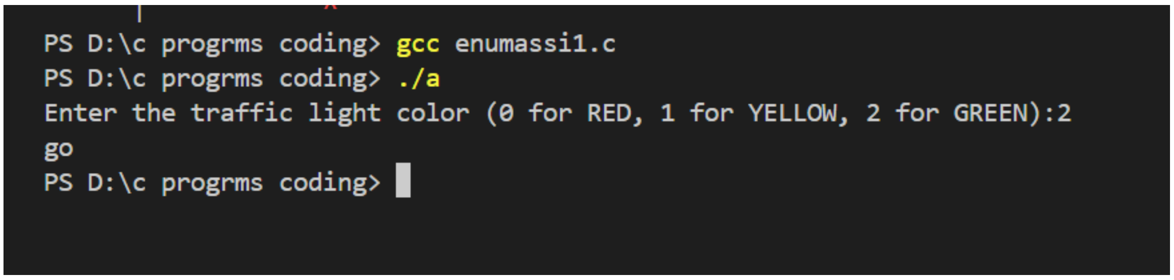
GREEN: "Go"



```c
#include<stdio.h>

enum trafficlight{
    RED=0,
    YELLOW=1,
    GREEN=2


};

int main(){
    enum trafficlight light;
    int input;

    printf("Enter the traffic light color (0 for RED, 1 for YELLOW, 2 for GREEN):");
    scanf("%d",&input);

    light=input;

    switch(light){
        case RED:
        printf("stop\n");
        break;
        case YELLOW:
        printf("Ready to move\n");
```



```c
    printf("Enter the traffic light color (0 for RED, 1 for YELLOW, 2 for GREEN): ");
    scanf("%d",&input);

    light=input;

    switch(light){
        case RED:
        printf("stop\n");
        break;
        case YELLOW:
        printf("Ready to move\n");
        break;
        case GREEN:
        printf("go\n");
        break;
        default:
        printf("invalid\n");
    }
```



```
PS D:\c progrms coding> gcc enumassi1.c
PS D:\c progrms coding> ./a
Enter the traffic light color (0 for RED, 1 for YELLOW, 2 for GREEN):2
go
PS D:\c progrms coding> 
```

6. Problem 2: Days of the Week

Problem Statement:

Write a C program that uses an enum to represent the days of the week. The program should:

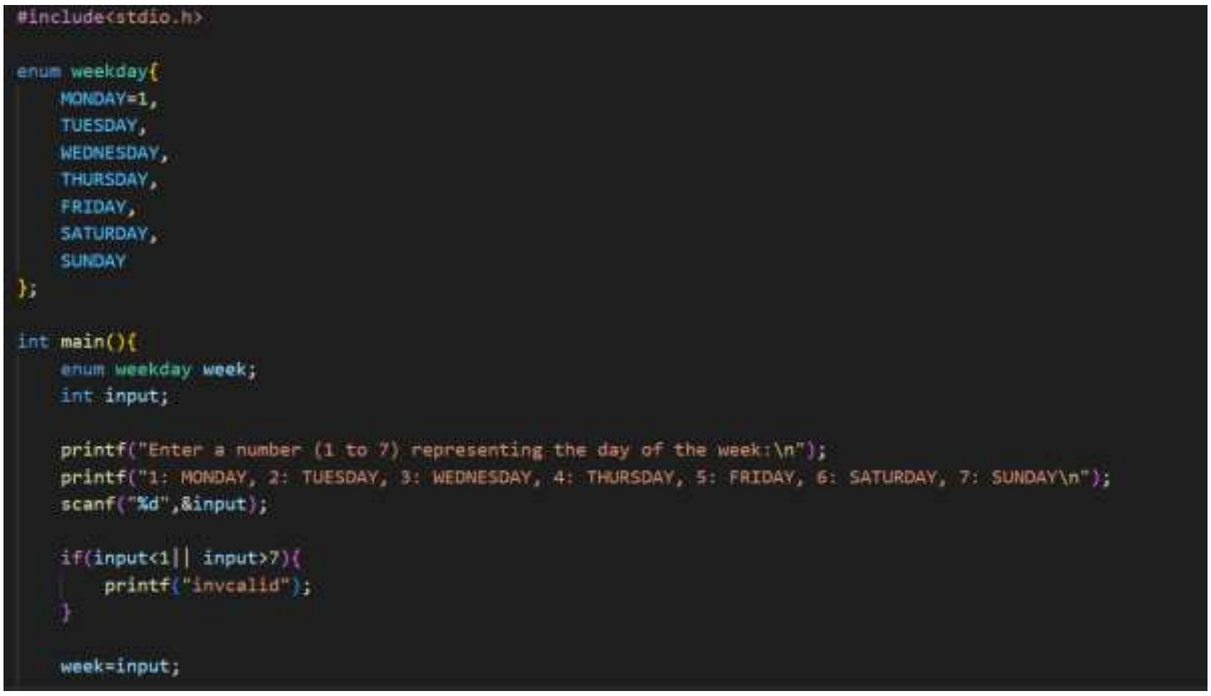
Define an enum named Weekday with values MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, and SUNDAY.

Accept a number (1 to 7) from the user representing the day of the week.

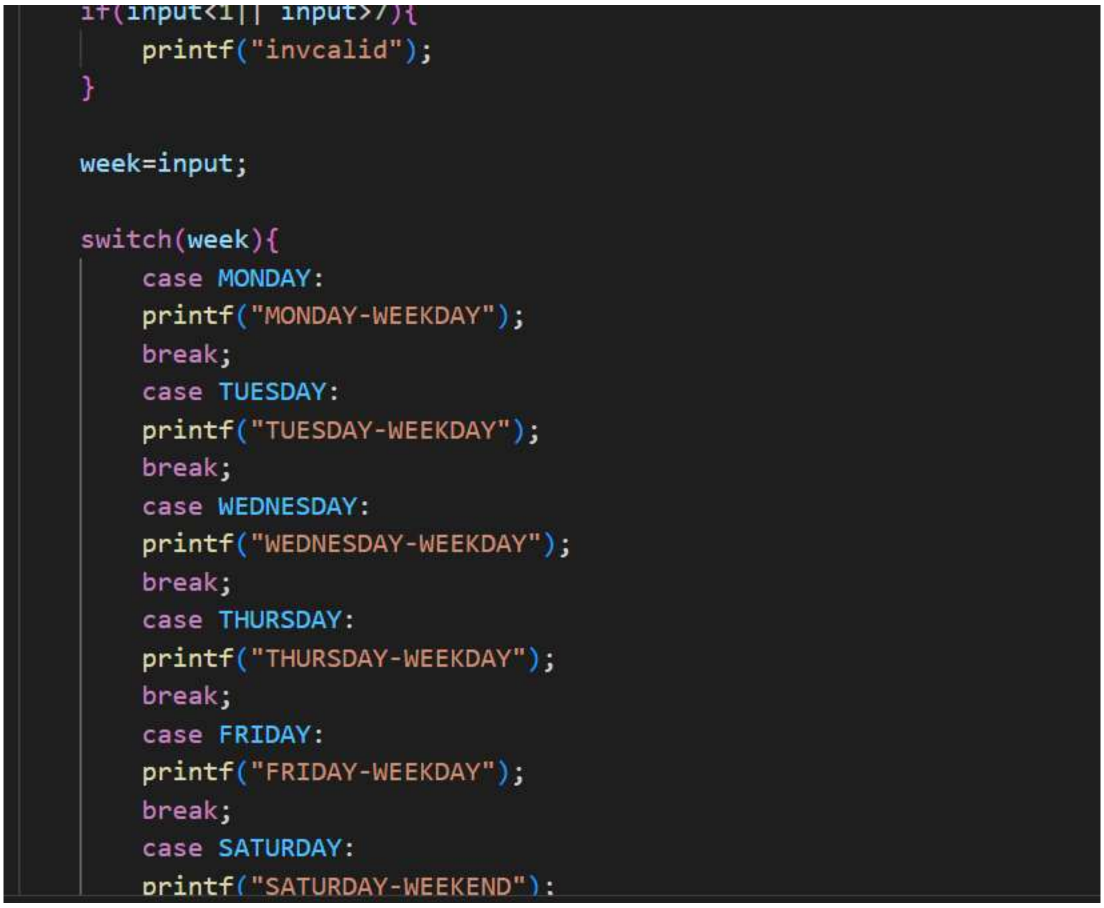
Print the name of the day and whether it is a weekday or a weekend.

Weekends: SATURDAY and SUNDAY

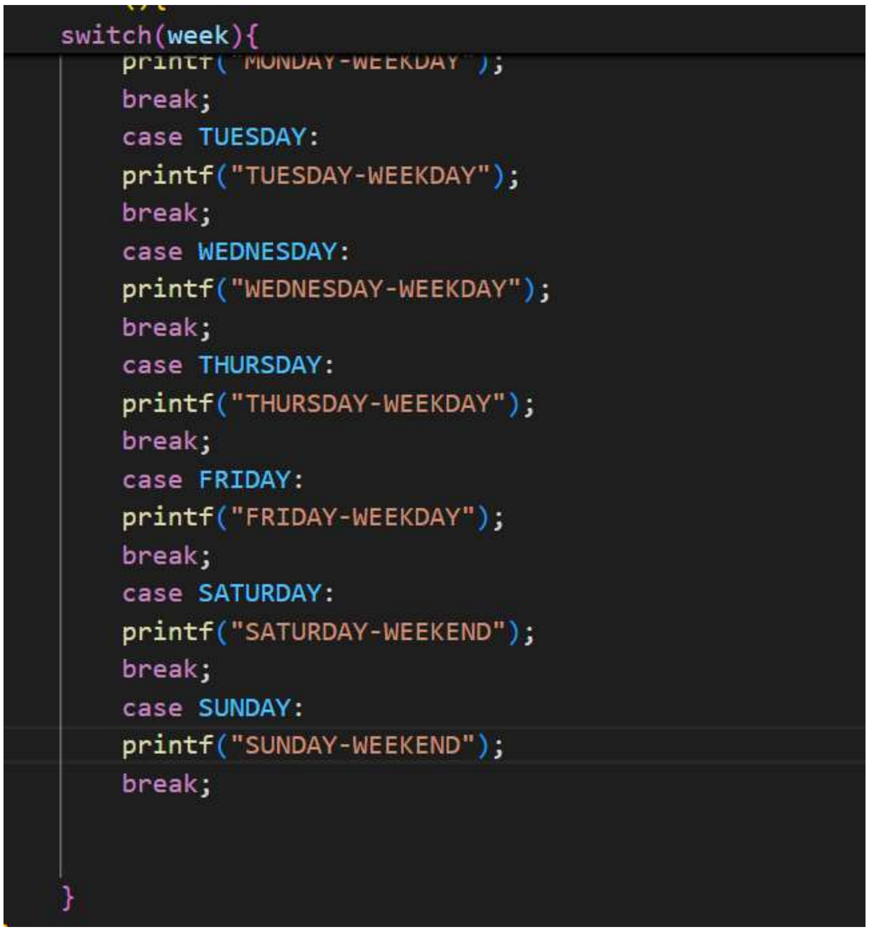
Weekdays: The rest



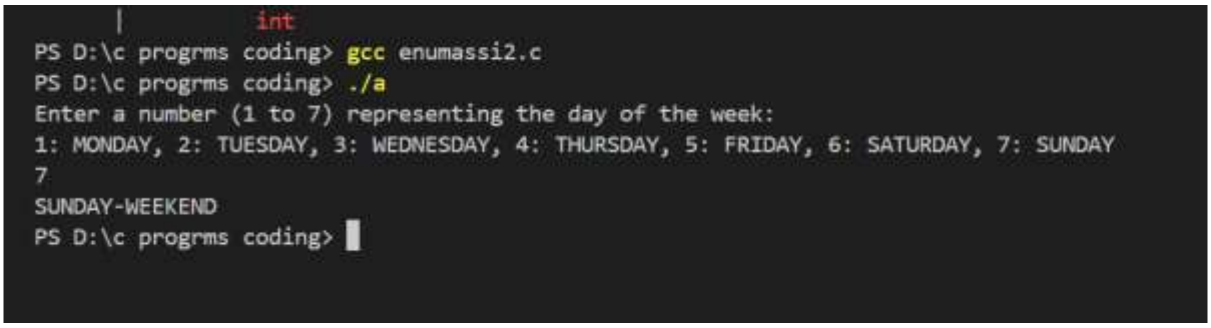
```c
#include<stdio.h>

enum weekday{
    MONDAY=1,
    TUESDAY,
    WEDNESDAY,
    THURSDAY,
    FRIDAY,
    SATURDAY,
    SUNDAY
};

int main(){
    enum weekday week;
    int input;

    printf("Enter a number (1 to 7) representing the day of the week:\n");
    printf("1: MONDAY, 2: TUESDAY, 3: WEDNESDAY, 4: THURSDAY, 5: FRIDAY, 6: SATURDAY, 7: SUNDAY\n");
    scanf("%d",&input);

    if(input<1|| input>7){
        printf("invcalid");
    }

    week=input;
```



```c
    if(input<1|| input>7){
        printf("invcalid");
    }

    week=input;

    switch(week){
        case MONDAY:
        printf("MONDAY-WEEKDAY");
        break;
        case TUESDAY:
        printf("TUESDAY-WEEKDAY");
        break;
        case WEDNESDAY:
        printf("WEDNESDAY-WEEKDAY");
        break;
        case THURSDAY:
        printf("THURSDAY-WEEKDAY");
        break;
        case FRIDAY:
        printf("FRIDAY-WEEKDAY");
        break;
        case SATURDAY:
        printf("SATURDAY-WEEKEND");
```



```
switch(week){
    printf("MONDAY-WEEKDAY");
    break;
    case TUESDAY:
    printf("TUESDAY-WEEKDAY");
    break;
    case WEDNESDAY:
    printf("WEDNESDAY-WEEKDAY");
    break;
    case THURSDAY:
    printf("THURSDAY-WEEKDAY");
    break;
    case FRIDAY:
    printf("FRIDAY-WEEKDAY");
    break;
    case SATURDAY:
    printf("SATURDAY-WEEKEND");
    break;
    case SUNDAY:
    printf("SUNDAY-WEEKEND");
    break;


}
```



```
PS D:\c progrms coding> gcc enumassi2.c
PS D:\c progrms coding> ./a
Enter a number (1 to 7) representing the day of the week:
1: MONDAY, 2: TUESDAY, 3: WEDNESDAY, 4: THURSDAY, 5: FRIDAY, 6: SATURDAY, 7: SUNDAY
7
SUNDAY-WEEKEND
PS D:\c progrms coding>
```

7. Problem 3: Shapes and Their Areas

Problem Statement:

Write a C program to calculate the area of a shape based on user input using enum. The program should:

Define an enum named Shape with values CIRCLE, RECTANGLE, and TRIANGLE.

Prompt the user to select a shape (0 for CIRCLE, 1 for RECTANGLE, 2 for TRIANGLE).

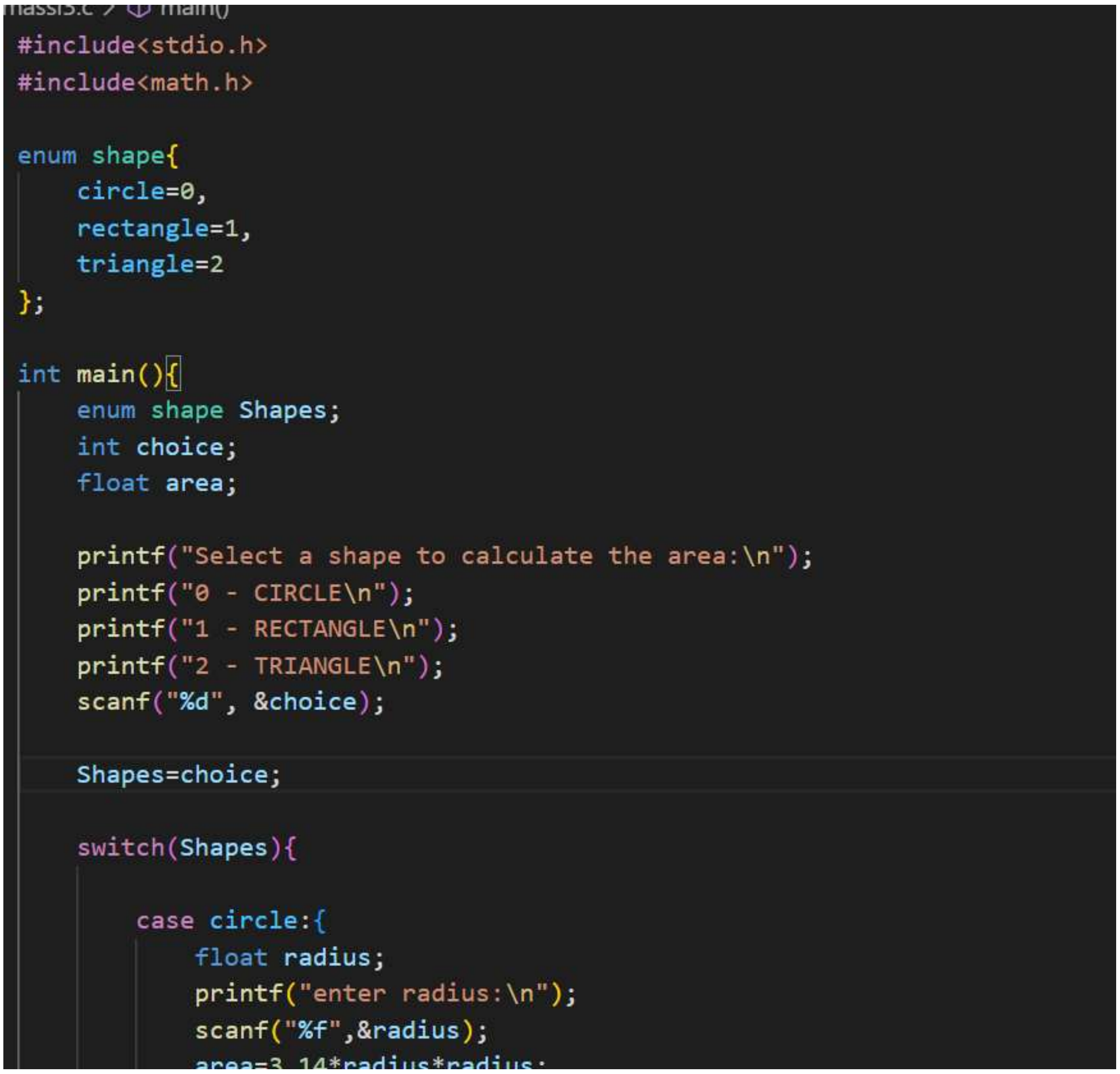
Based on the selection, input the required dimensions:

For CIRCLE: Radius

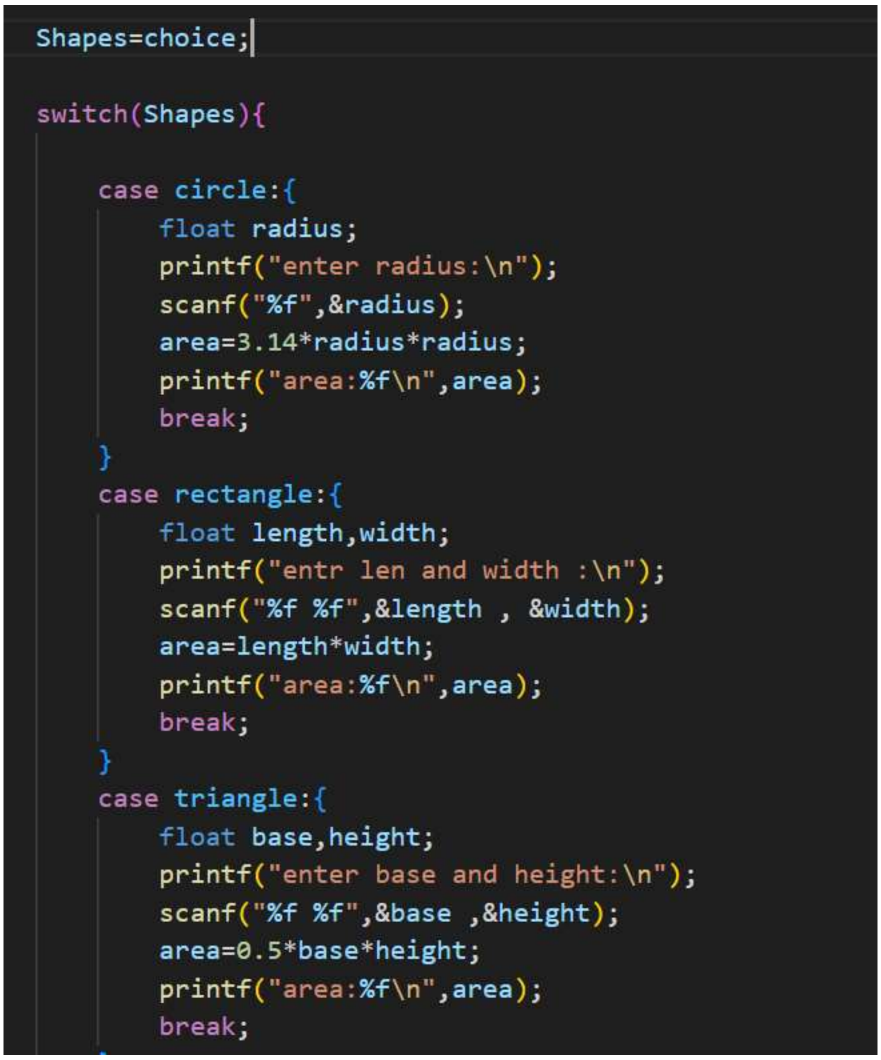
For RECTANGLE: Length and breadth

For TRIANGLE: Base and height

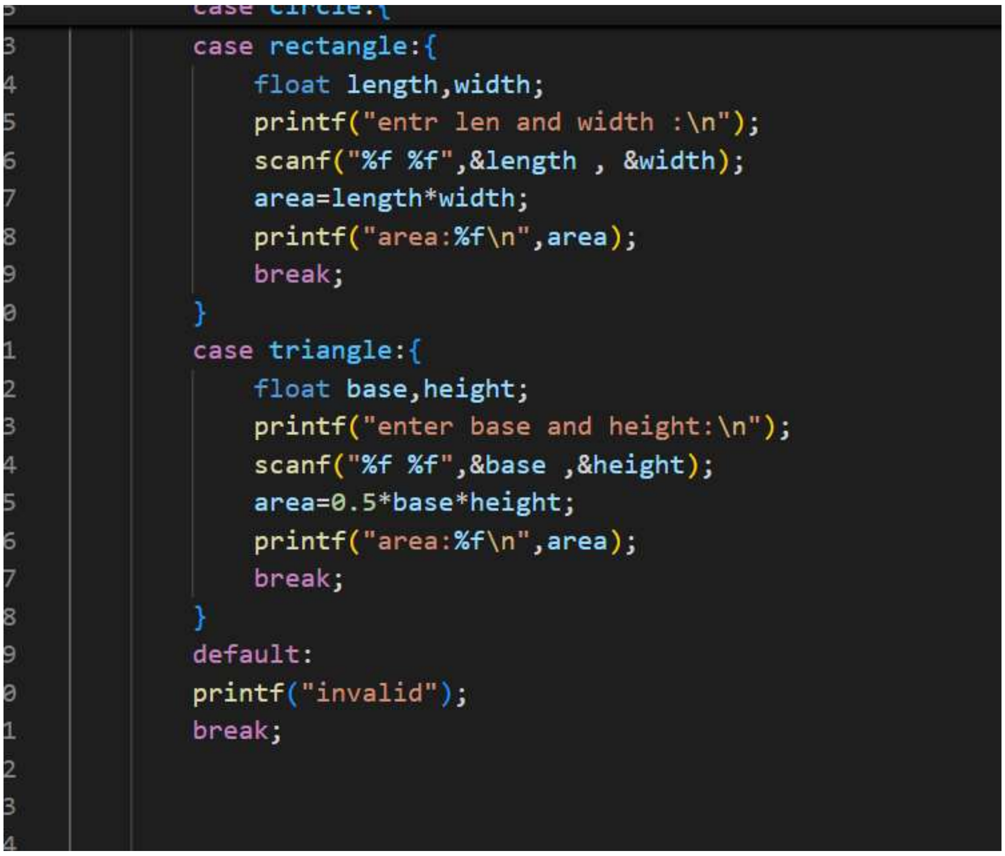
Calculate and display the area of the selected shape



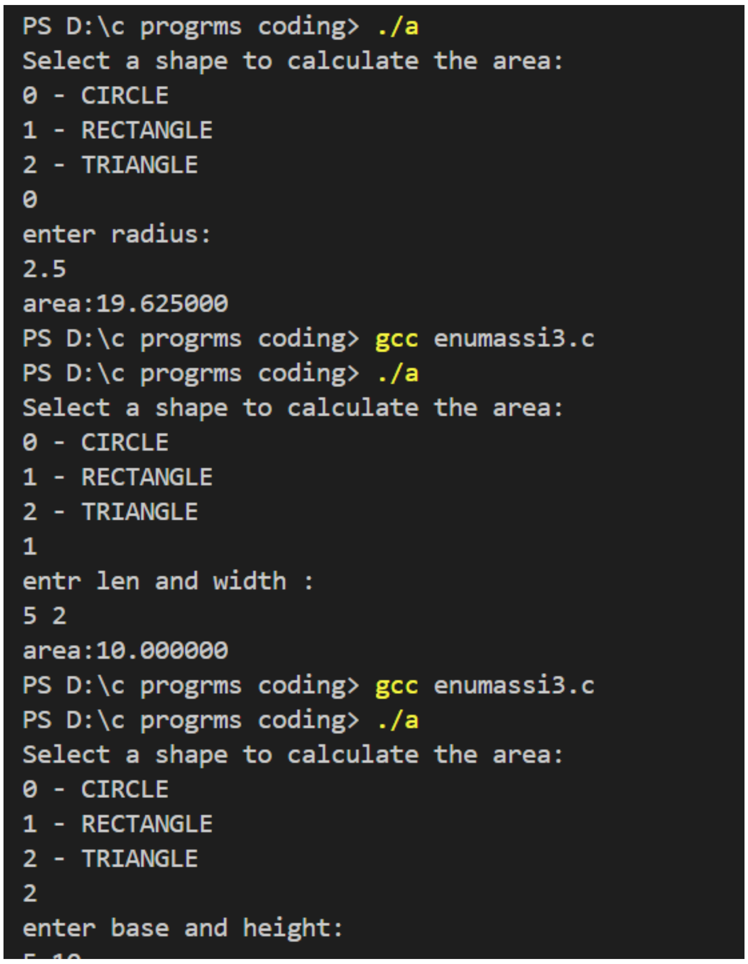
```c
#include<stdio.h>
#include<math.h>

enum shape{
    circle=0,
    rectangle=1,
    triangle=2
};

int main(){
    enum shape Shapes;
    int choice;
    float area;

    printf("Select a shape to calculate the area:\n");
    printf("0 - CIRCLE\n");
    printf("1 - RECTANGLE\n");
    printf("2 - TRIANGLE\n");
    scanf("%d", &choice);

    Shapes=choice;

    switch(Shapes){

        case circle:{
            float radius;
            printf("enter radius:\n");
            scanf("%f",&radius);
            area=3.14*radius*radius;
```



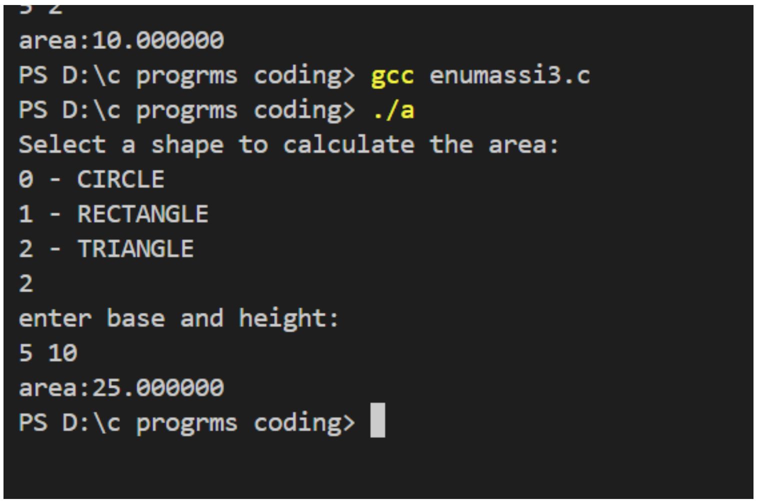
```
Shapes=choice;

switch(Shapes){

    case circle:{
        float radius;
        printf("enter radius:\n");
        scanf("%f",&radius);
        area=3.14*radius*radius;
        printf("area:%f\n",area);
        break;
    }
    case rectangle:{
        float length,width;
        printf("entr len and width :\n");
        scanf("%f %f",&length , &width);
        area=length*width;
        printf("area:%f\n",area);
        break;
    }
    case triangle:{
        float base,height;
        printf("enter base and height:\n");
        scanf("%f %f",&base ,&height);
        area=0.5*base*height;
        printf("area:%f\n",area);
        break;
```



```
        case rectangle:{
            float length,width;
            printf("entr len and width :\n");
            scanf("%f %f",&length , &width);
            area=length*width;
            printf("area:%f\n",area);
            break;
        }
        case triangle:{
            float base,height;
            printf("enter base and height:\n");
            scanf("%f %f",&base ,&height);
            area=0.5*base*height;
            printf("area:%f\n",area);
            break;
        }
        default:
        printf("invalid");
        break;
```



```
PS D:\c progrms coding> ./a
Select a shape to calculate the area:
0 - CIRCLE
1 - RECTANGLE
2 - TRIANGLE
0
enter radius:
2.5
area:19.625000
PS D:\c progrms coding> gcc enumassi3.c
PS D:\c progrms coding> ./a
Select a shape to calculate the area:
0 - CIRCLE
1 - RECTANGLE
2 - TRIANGLE
1
entr len and width :
5 2
area:10.000000
PS D:\c progrms coding> gcc enumassi3.c
PS D:\c progrms coding> ./a
Select a shape to calculate the area:
0 - CIRCLE
1 - RECTANGLE
2 - TRIANGLE
2
enter base and height:
```



```
5 2
area:10.000000
PS D:\c progrms coding> gcc enumassi3.c
PS D:\c progrms coding> ./a
Select a shape to calculate the area:
0 - CIRCLE
1 - RECTANGLE
2 - TRIANGLE
2
enter base and height:
5 10
area:25.000000
PS D:\c progrms coding>
```

8.Problem 5: User Roles in a System

Problem Statement:

Write a C program to define user roles in a system using enum. The program should:

Define an enum named UserRole with values ADMIN, EDITOR, VIEWER, and GUEST.

Accept the user role as input (0 for ADMIN, 1 for EDITOR, etc.).

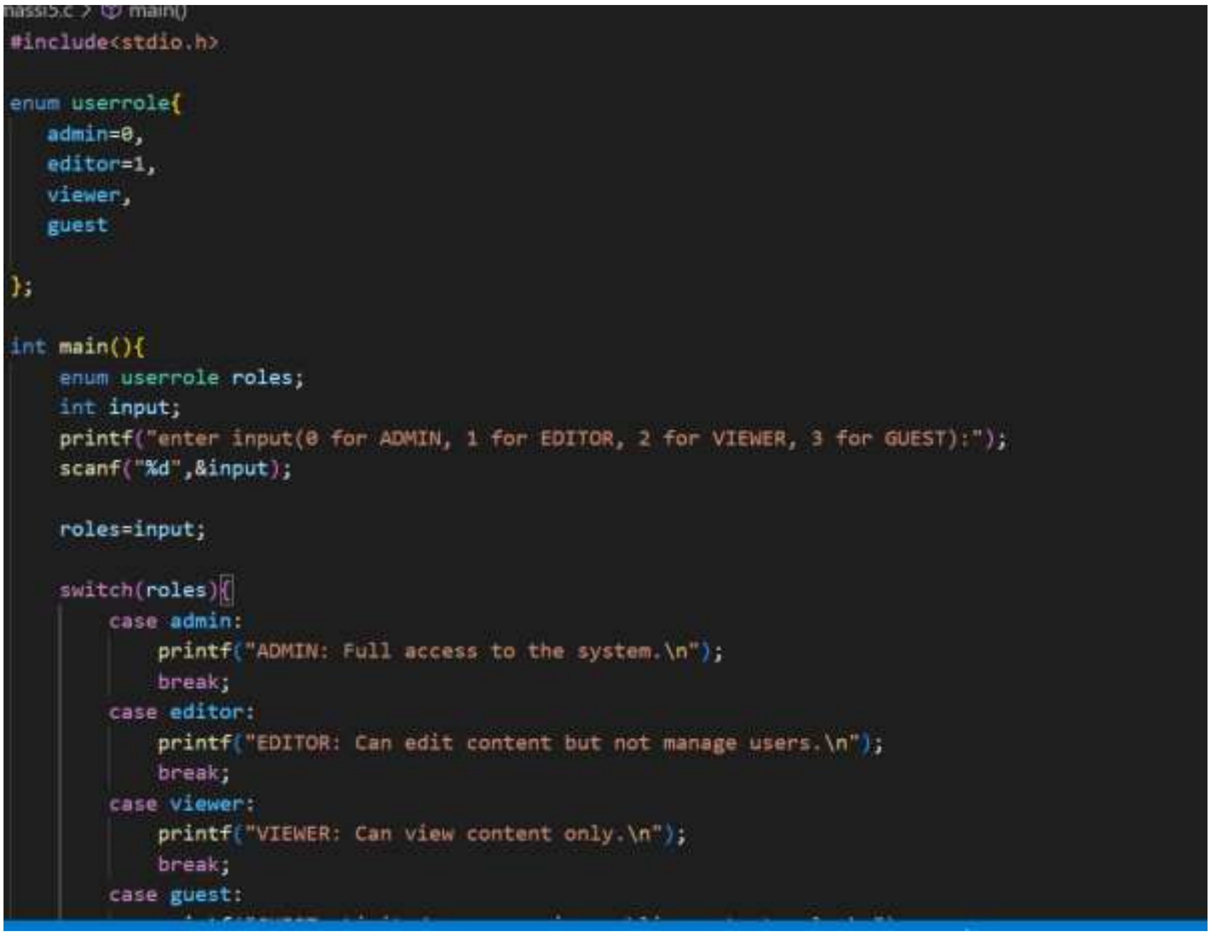
Display the permissions associated with each role:

ADMIN: "Full access to the system."

EDITOR: "Can edit content but not manage users."

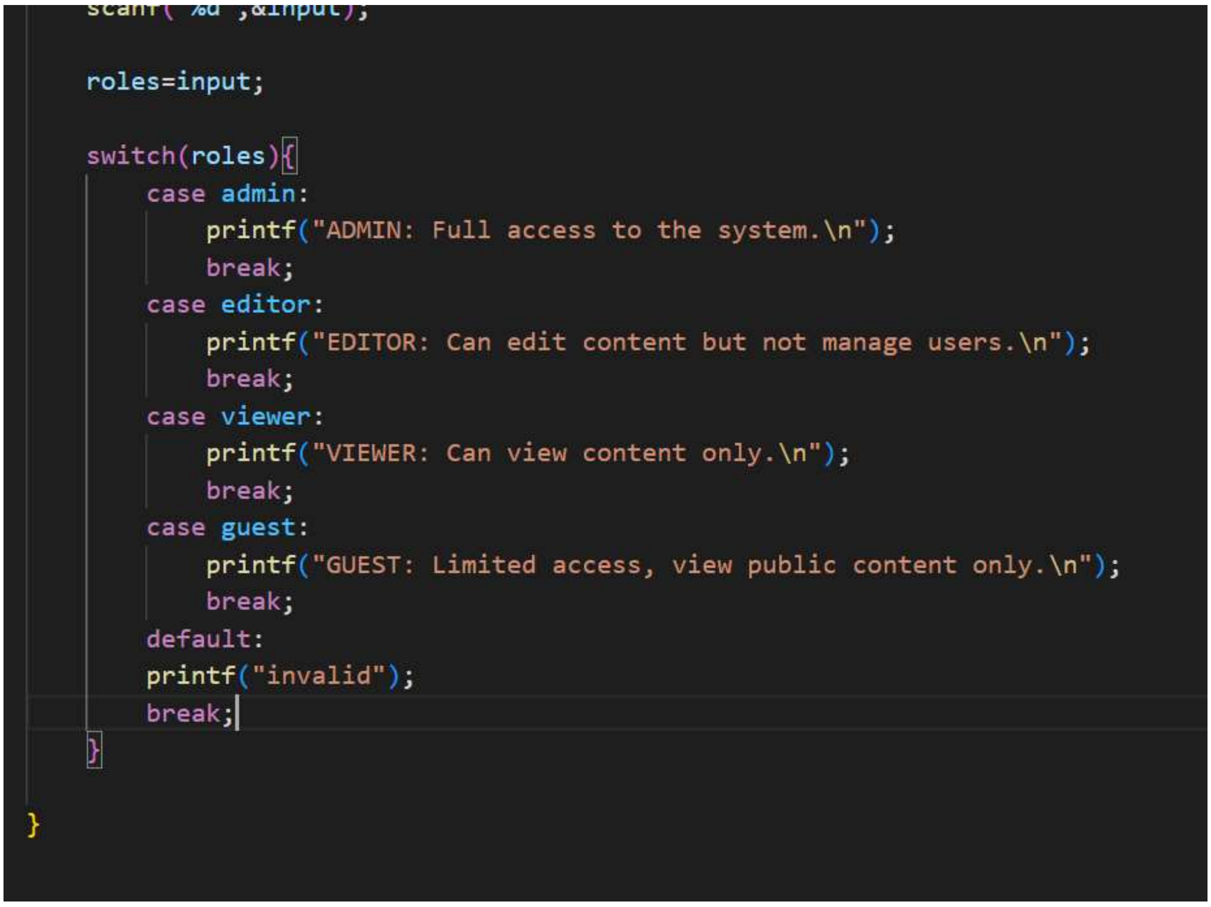
VIEWER: "Can view content only."

GUEST: "Limited access, view public content only."

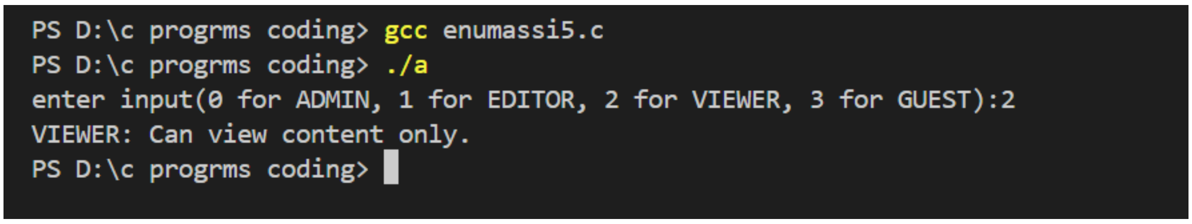


nassi5.c > main()

```
#include<stdio.h>

enum userrole{
   admin=0,
   editor=1,
   viewer,
   guest

};

int main(){
    enum userrole roles;
    int input;
    printf("enter input(0 for ADMIN, 1 for EDITOR, 2 for VIEWER, 3 for GUEST):");
    scanf("%d",&input);

    roles=input;

    switch(roles){
        case admin:
            printf("ADMIN: Full access to the system.\n");
            break;
        case editor:
            printf("EDITOR: Can edit content but not manage users.\n");
            break;
        case viewer:
            printf("VIEWER: Can view content only.\n");
            break;
        case guest:
```



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

    roles=input;

    switch(roles){
        case admin:
            printf("ADMIN: Full access to the system.\n");
            break;
        case editor:
            printf("EDITOR: Can edit content but not manage users.\n");
            break;
        case viewer:
            printf("VIEWER: Can view content only.\n");
            break;
        case guest:
            printf("GUEST: Limited access, view public content only.\n");
            break;
        default:
        printf("invalid");
        break;
    }

}
```



```
PS D:\c progrms coding> gcc enumassi5.c
PS D:\c progrms coding> ./a
enter input(0 for ADMIN, 1 for EDITOR, 2 for VIEWER, 3 for GUEST):2
VIEWER: Can view content only.
PS D:\c progrms coding>
```

9. Problem 4: Error Codes in a Program

Problem Statement:

Write a C program to simulate error handling using enum. The program should:

Define an enum named ErrorCode with values:

SUCCESS (0)

FILE_NOT_FOUND (1)

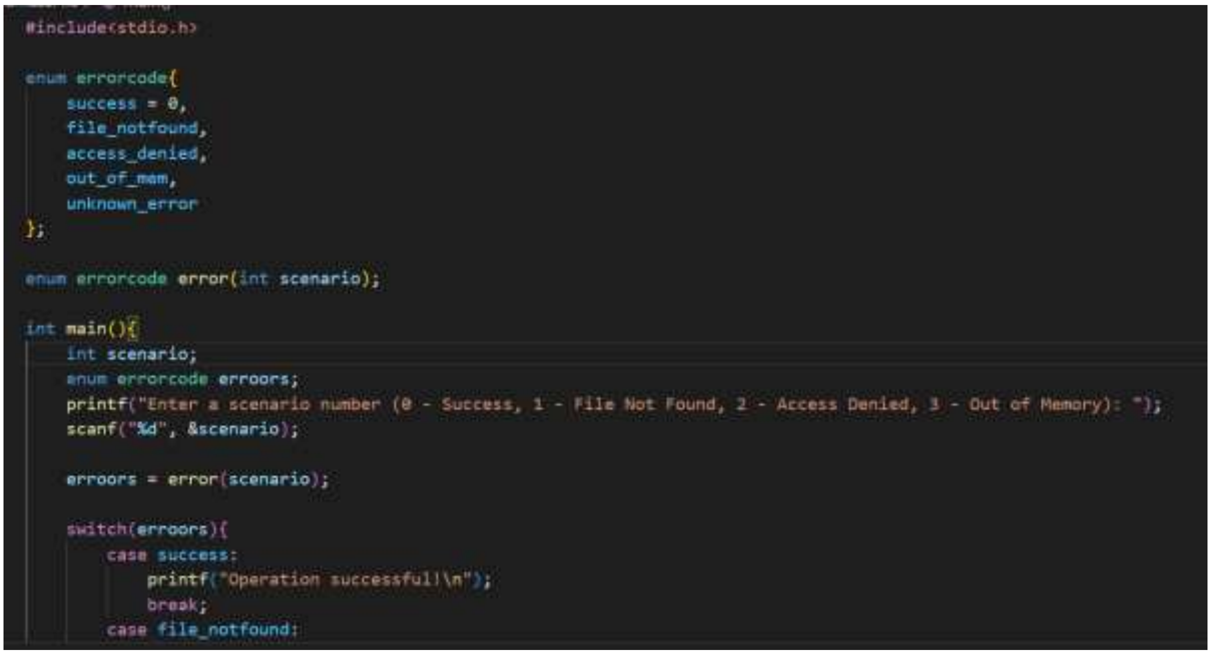
ACCESS_DENIED (2)

OUT_OF_MEMORY (3)

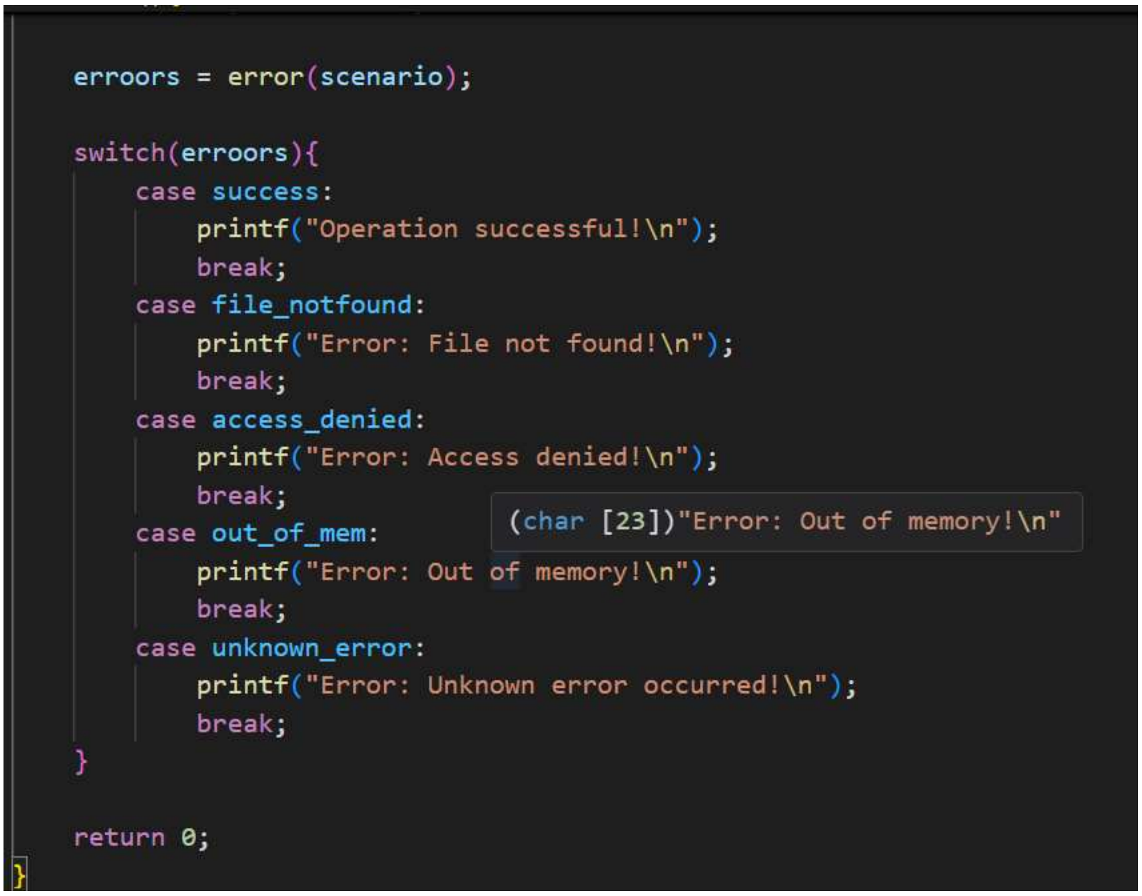
UNKNOWN_ERROR (4)

Simulate a function that returns an error code based on a scenario.

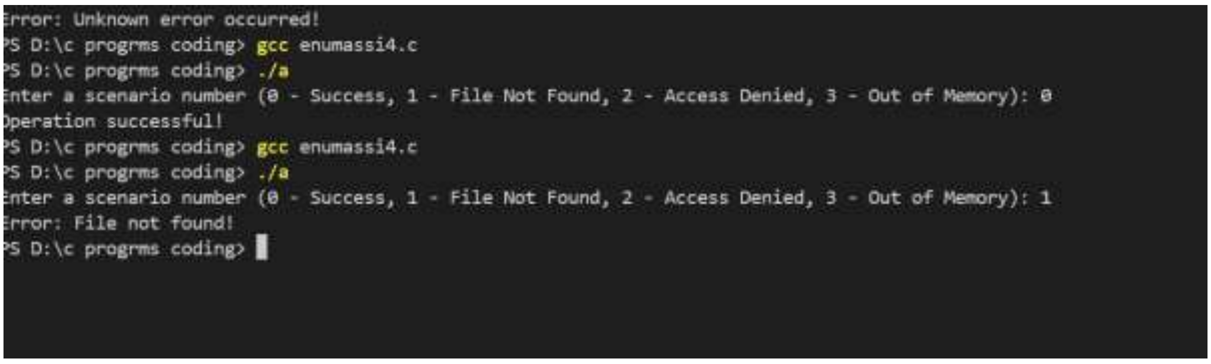
Based on the returned error code, print an appropriate message to the user



```c
#include<stdio.h>

enum errorcode{
    success = 0,
    file_notfound,
    access_denied,
    out_of_mem,
    unknown_error
};

enum errorcode error(int scenario);

int main(){
    int scenario;
    enum errorcode erroors;
    printf("Enter a scenario number (0 - Success, 1 - File Not Found, 2 - Access Denied, 3 - Out of Memory): ");
    scanf("%d", &scenario);

    erroors = error(scenario);

    switch(erroors){
        case success:
            printf("Operation successful!\n");
            break;
        case file_notfound:
```



```
    erroors = error(scenario);

    switch(erroors){
        case success:
            printf("Operation successful!\n");
            break;
        case file_notfound:
            printf("Error: File not found!\n");
            break;
        case access_denied:
            printf("Error: Access denied!\n");
            break;
        case out_of_mem:                (char [23])"Error: Out of memory!\n"
            printf("Error: Out of memory!\n");
            break;
        case unknown_error:
            printf("Error: Unknown error occurred!\n");
            break;
    }

    return 0;
}
```



```
Error: Unknown error occurred!
PS D:\c progrms coding> gcc enumassi4.c
PS D:\c progrms coding> ./a
Enter a scenario number (0 - Success, 1 - File Not Found, 2 - Access Denied, 3 - Out of Memory): 0
Operation successful!
PS D:\c progrms coding> gcc enumassi4.c
PS D:\c progrms coding> ./a
Enter a scenario number (0 - Success, 1 - File Not Found, 2 - Access Denied, 3 - Out of Memory): 1
Error: File not found!
PS D:\c progrms coding>
```

10. Problem 1: Compact Date Storage

Problem Statement:

Write a C program to store and display dates using bit-fields. The program should:

Define a structure named Date with bit-fields:

day (5 bits): Stores the day of the month (1-31).

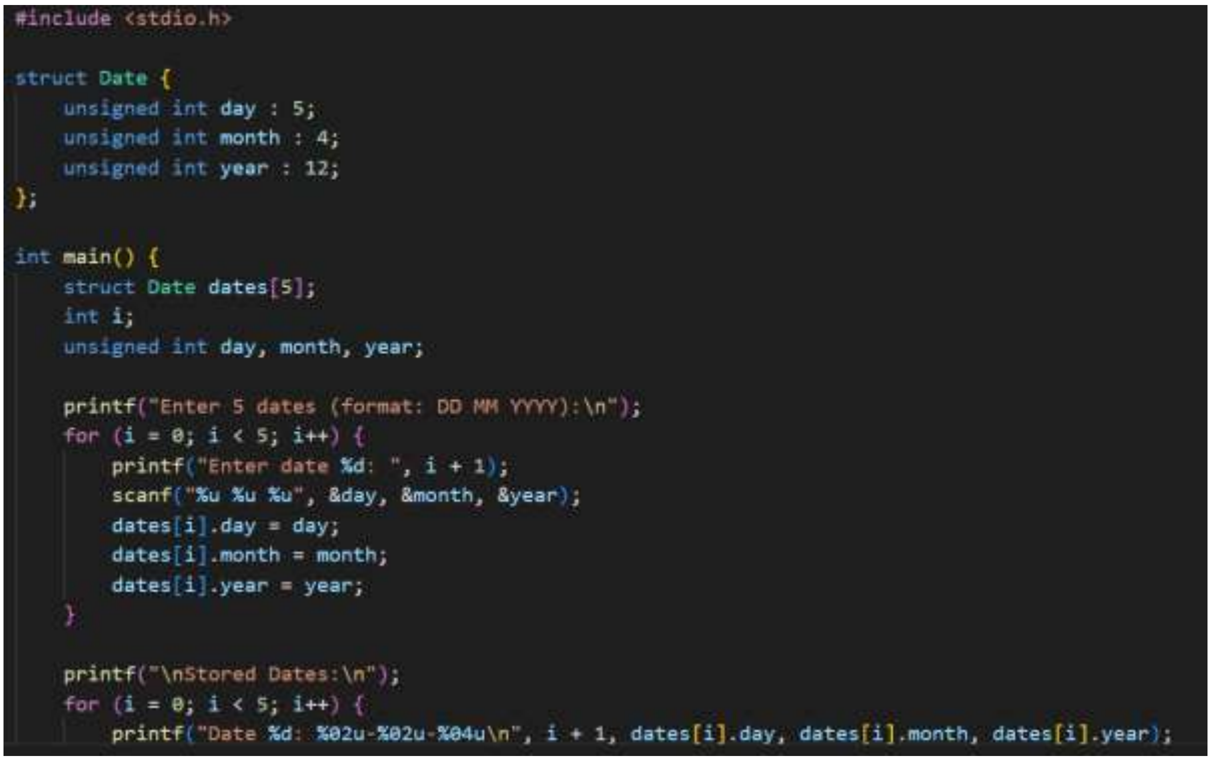
month (4 bits): Stores the month (1-12).

year (12 bits): Stores the year (e.g., 2024).

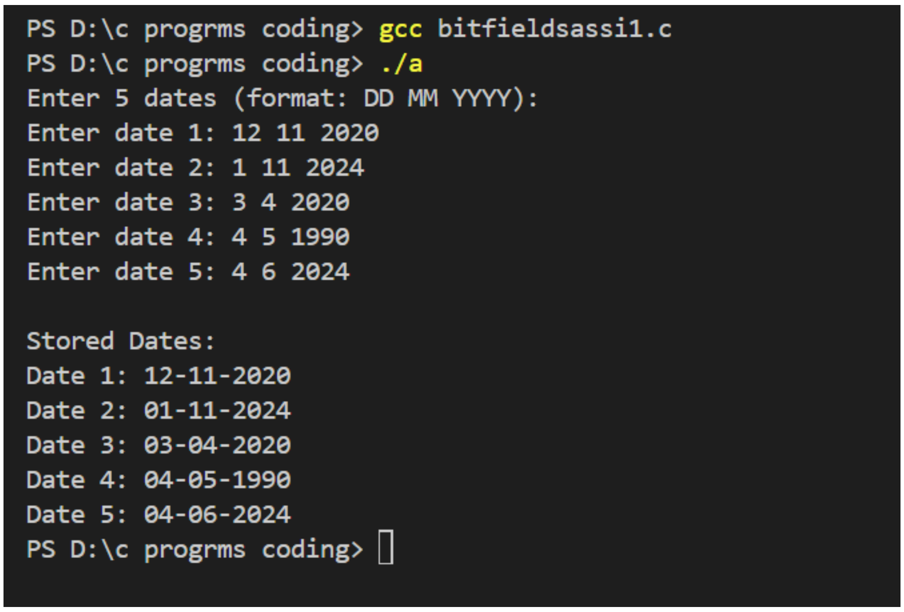
Create an array of dates to store 5 different dates.

Allow the user to input 5 dates in the format DD MM YYYY and store them in the array.

Display the stored dates in the format DD-MM-YYYY



```c
#include <stdio.h>

struct Date {
    unsigned int day : 5;
    unsigned int month : 4;
    unsigned int year : 12;
};

int main() {
    struct Date dates[5];
    int i;
    unsigned int day, month, year;

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

    printf("\nStored Dates:\n");
    for (i = 0; i < 5; i++) {
        printf("Date %d: %02u-%02u-%04u\n", i + 1, dates[i].day, dates[i].month, dates[i].year);
```



```
PS D:\c progrms coding> gcc bitfieldsassi1.c
PS D:\c progrms coding> ./a
Enter 5 dates (format: DD MM YYYY):
Enter date 1: 12 11 2020
Enter date 2: 1 11 2024
Enter date 3: 3 4 2020
Enter date 4: 4 5 1990
Enter date 5: 4 6 2024

Stored Dates:
Date 1: 12-11-2020
Date 2: 01-11-2024
Date 3: 03-04-2020
Date 4: 04-05-1990
Date 5: 04-06-2024
PS D:\c progrms coding>
```

11.

Problem 2: Status Flags for a Device

Problem Statement:

Write a C program to manage the status of a device using bit-fields. The program should:

Define a structure named DeviceStatus with the following bit-fields:

power (1 bit): 1 if the device is ON, 0 if OFF.

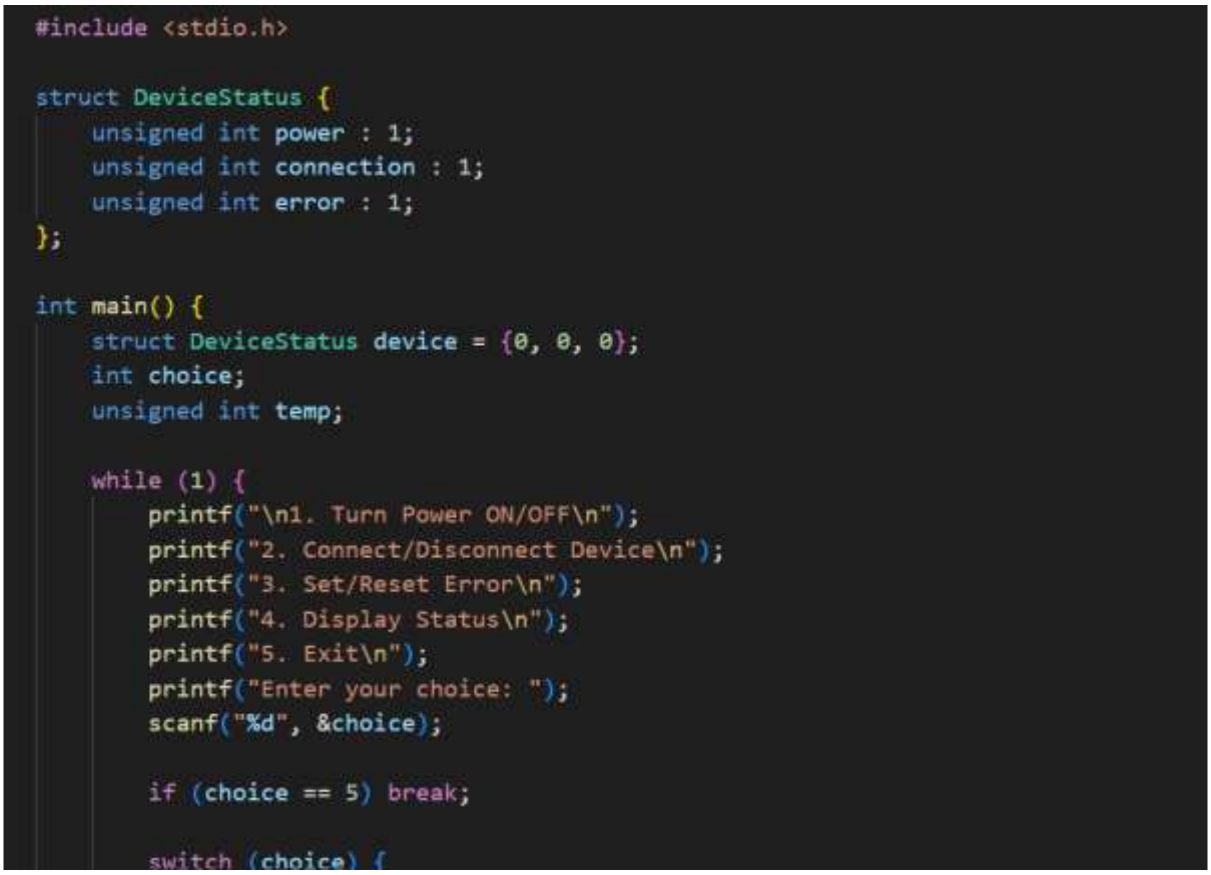
connection (1 bit): 1 if the device is connected, 0 if disconnected.

error (1 bit): 1 if there's an error, 0 otherwise.

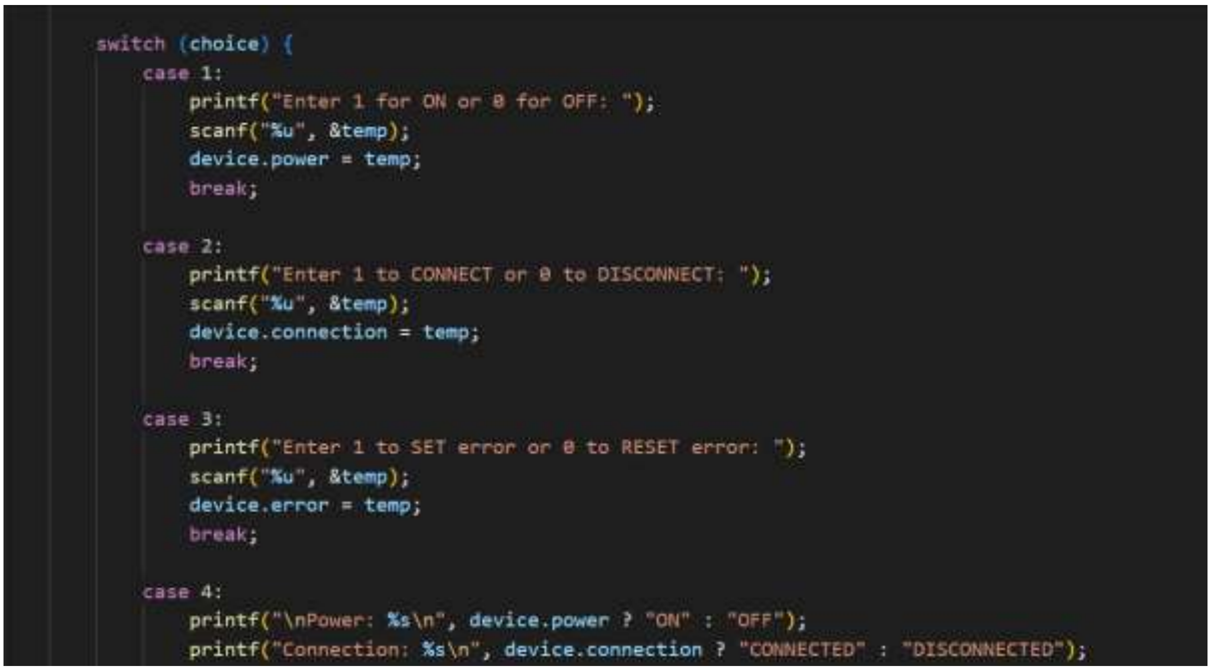
Simulate the device status by updating the bit-fields based on user input:

Allow the user to set or reset each status.

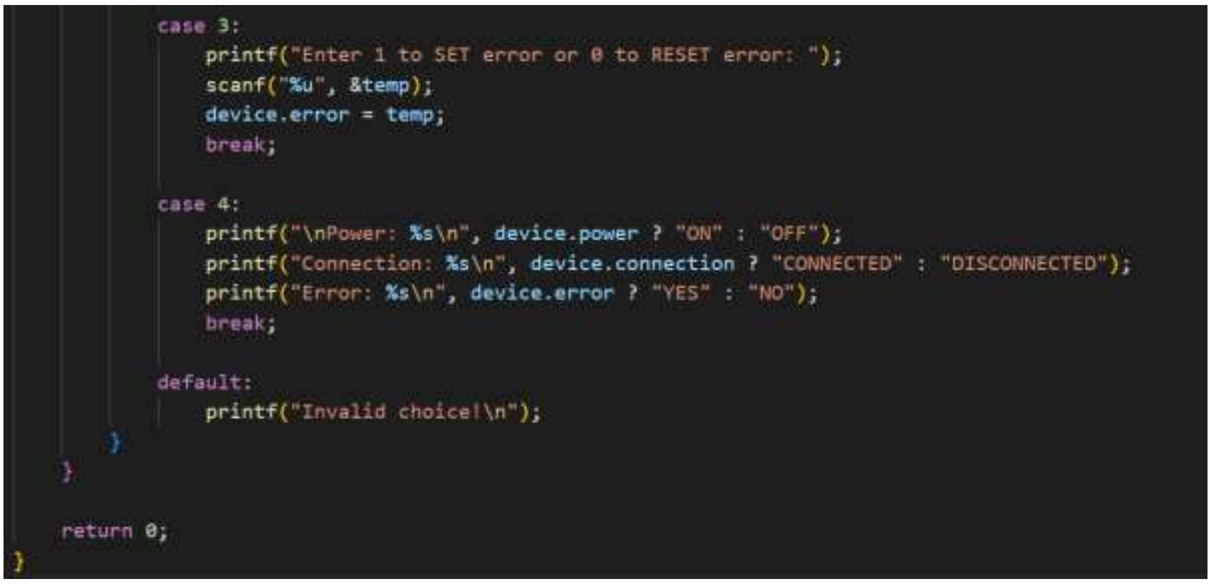
Display the current status of the device in a readable format (e.g., Power: ON, Connection: DISCONNECTED, Error: NO).



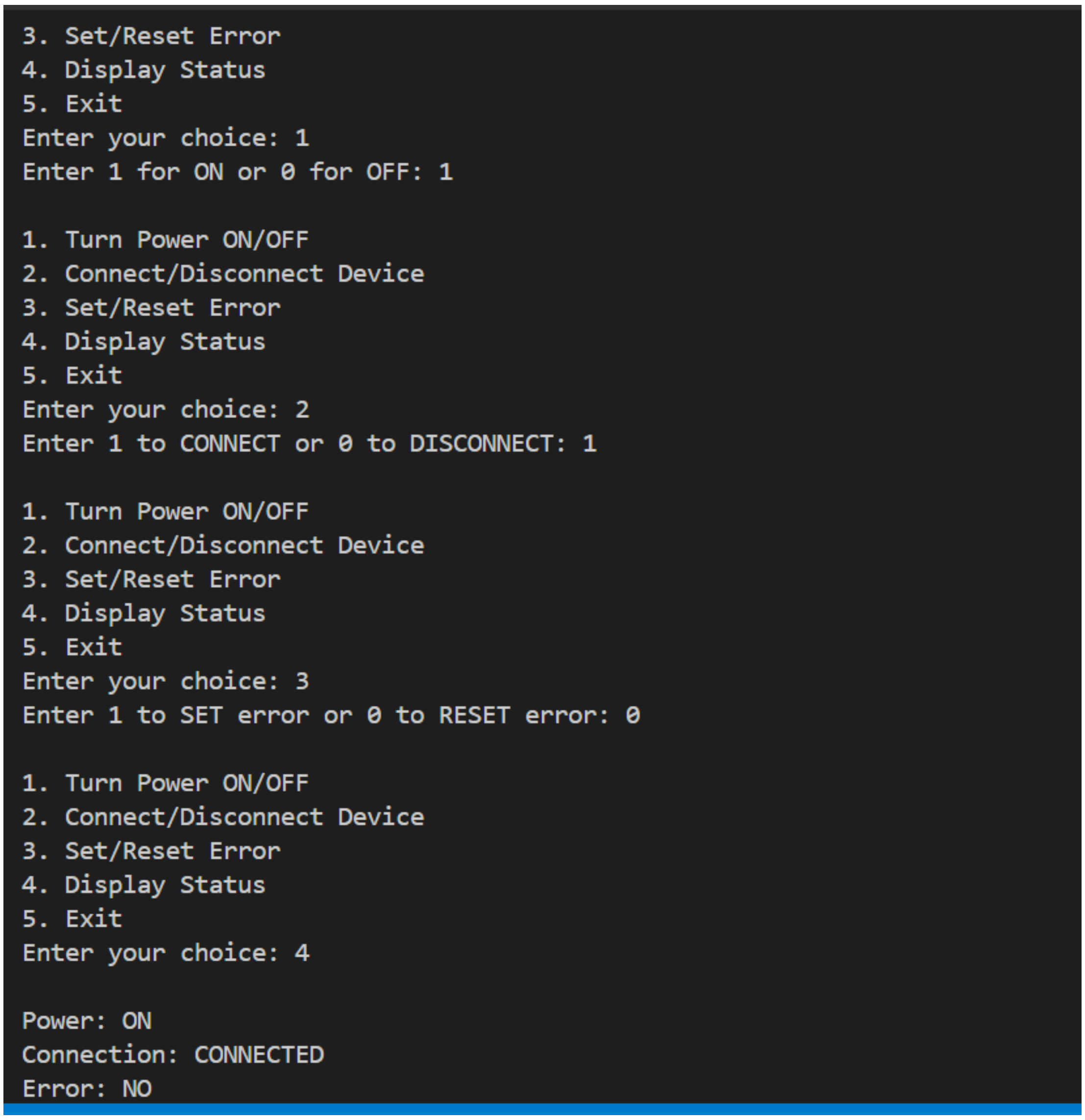
```c
#include <stdio.h>

struct DeviceStatus {
    unsigned int power : 1;
    unsigned int connection : 1;
    unsigned int error : 1;
};

int main() {
    struct DeviceStatus device = {0, 0, 0};
    int choice;
    unsigned int temp;

    while (1) {
        printf("\n1. Turn Power ON/OFF\n");
        printf("2. Connect/Disconnect Device\n");
        printf("3. Set/Reset Error\n");
        printf("4. Display Status\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        if (choice == 5) break;

        switch (choice) {
```



```c
        switch (choice) {
            case 1:
                printf("Enter 1 for ON or 0 for OFF: ");
                scanf("%u", &temp);
                device.power = temp;
                break;

            case 2:
                printf("Enter 1 to CONNECT or 0 to DISCONNECT: ");
                scanf("%u", &temp);
                device.connection = temp;
                break;

            case 3:
                printf("Enter 1 to SET error or 0 to RESET error: ");
                scanf("%u", &temp);
                device.error = temp;
                break;

            case 4:
                printf("\nPower: %s\n", device.power ? "ON" : "OFF");
                printf("Connection: %s\n", device.connection ? "CONNECTED" : "DISCONNECTED");
```



```c
            case 3:
                printf("Enter 1 to SET error or 0 to RESET error: ");
                scanf("%u", &temp);
                device.error = temp;
                break;

            case 4:
                printf("\nPower: %s\n", device.power ? "ON" : "OFF");
                printf("Connection: %s\n", device.connection ? "CONNECTED" : "DISCONNECTED");
                printf("Error: %s\n", device.error ? "YES" : "NO");
                break;

            default:
                printf("Invalid choice!\n");
        }
    }

    return 0;
}
```



```
3. Set/Reset Error
4. Display Status
5. Exit
Enter your choice: 1
Enter 1 for ON or 0 for OFF: 1

1. Turn Power ON/OFF
2. Connect/Disconnect Device
3. Set/Reset Error
4. Display Status
5. Exit
Enter your choice: 2
Enter 1 to CONNECT or 0 to DISCONNECT: 1

1. Turn Power ON/OFF
2. Connect/Disconnect Device
3. Set/Reset Error
4. Display Status
5. Exit
Enter your choice: 3
Enter 1 to SET error or 0 to RESET error: 0

1. Turn Power ON/OFF
2. Connect/Disconnect Device
3. Set/Reset Error
4. Display Status
5. Exit
Enter your choice: 4

Power: ON
Connection: CONNECTED
Error: NO
```

12. Problem 3: Storage Permissions

Problem Statement:

Write a C program to represent file permissions using bit-fields. The program should:

Define a structure named FilePermissions with the following bit-fields:

read (1 bit): Permission to read the file.

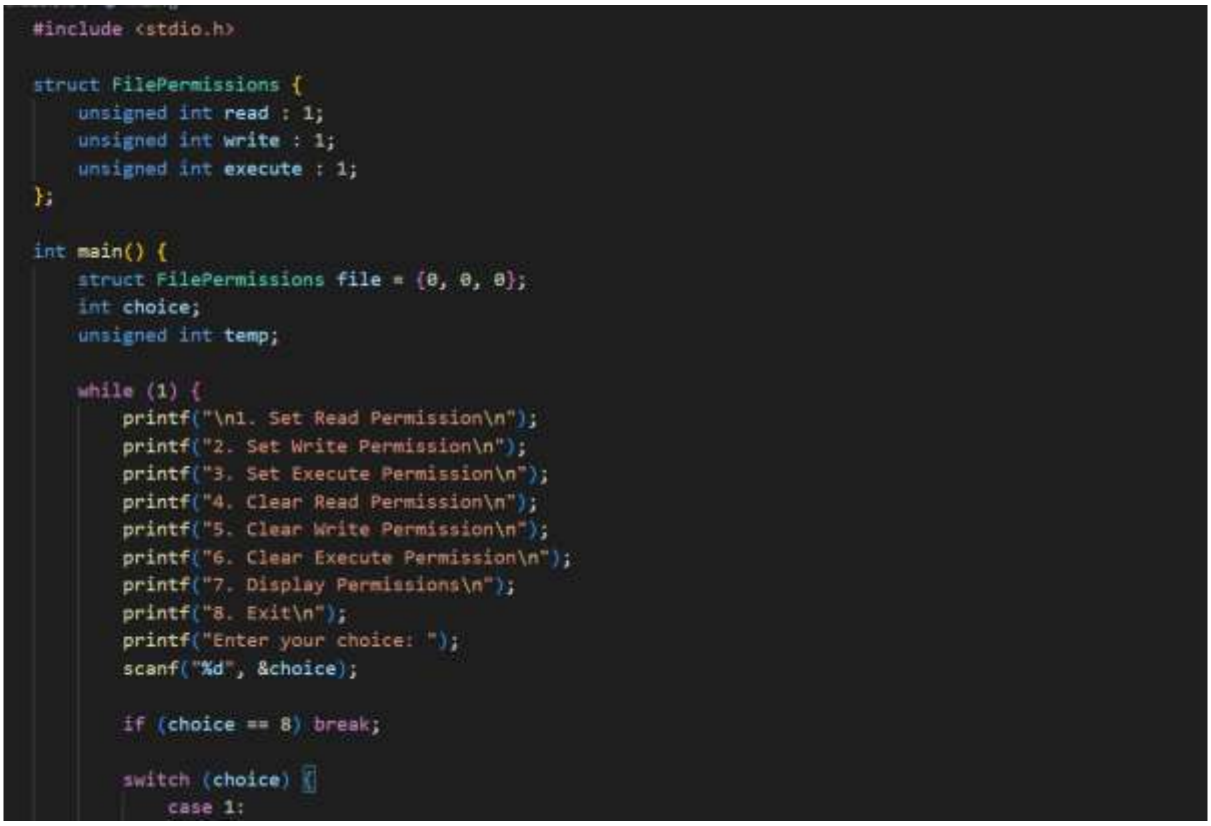
write (1 bit): Permission to write to the file.

execute (1 bit): Permission to execute the file.

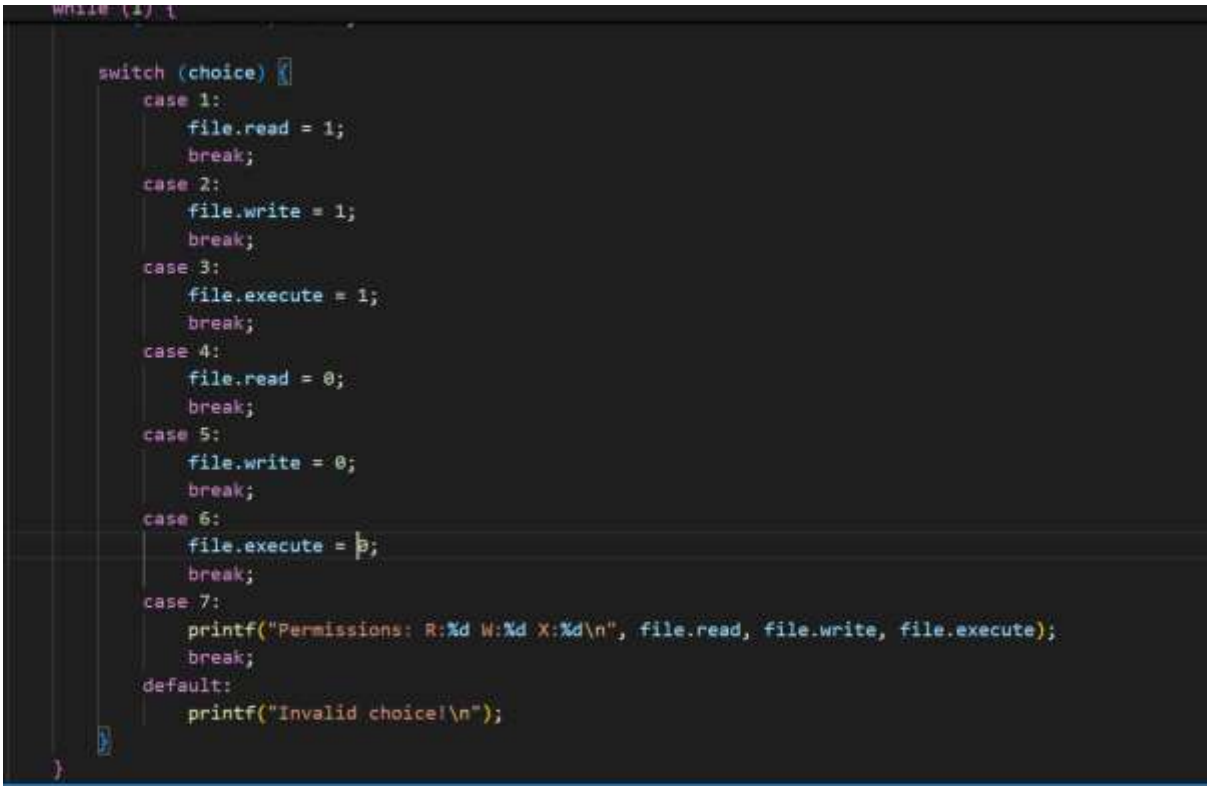
Simulate managing file permissions:

Allow the user to set or clear each permission for a file.

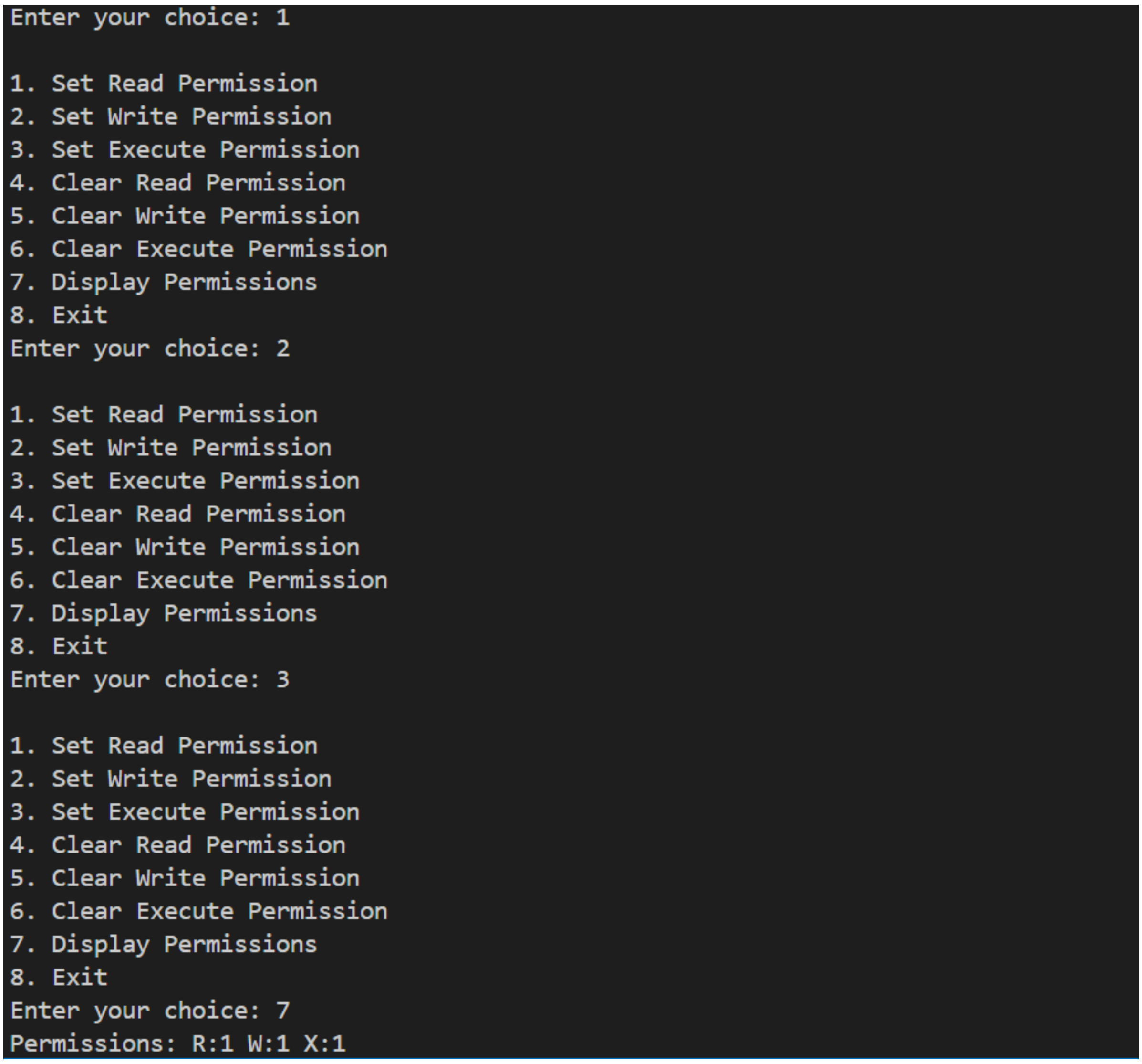
Display the current permissions in the format R:1 W:0 X:1 (1 for permission granted, 0 for denied)

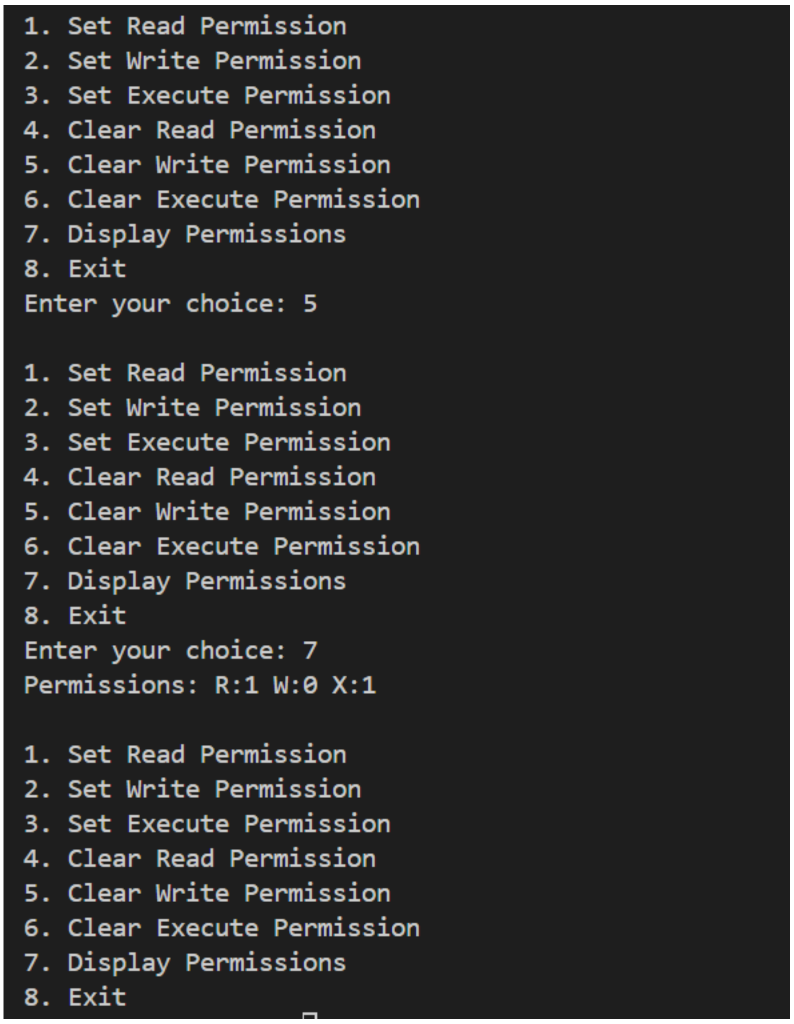


```
#include <stdio.h>

struct FilePermissions {
    unsigned int read : 1;
    unsigned int write : 1;
    unsigned int execute : 1;
};

int main() {
    struct FilePermissions file = {0, 0, 0};
    int choice;
    unsigned int temp;

    while (1) {
        printf("\n1. Set Read Permission\n");
        printf("2. Set Write Permission\n");
        printf("3. Set Execute Permission\n");
        printf("4. Clear Read Permission\n");
        printf("5. Clear Write Permission\n");
        printf("6. Clear Execute Permission\n");
        printf("7. Display Permissions\n");
        printf("8. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        if (choice == 8) break;

        switch (choice) {
            case 1:
```



```c
while (1) {

    switch (choice) {
        case 1:
            file.read = 1;
            break;
        case 2:
            file.write = 1;
            break;
        case 3:
            file.execute = 1;
            break;
        case 4:
            file.read = 0;
            break;
        case 5:
            file.write = 0;
            break;
        case 6:
            file.execute = 0;
            break;
        case 7:
            printf("Permissions: R:%d W:%d X:%d\n", file.read, file.write, file.execute);
            break;
        default:
            printf("Invalid choice!\n");
    }
}
```


```
Enter your choice: 1

1. Set Read Permission
2. Set Write Permission
3. Set Execute Permission
4. Clear Read Permission
5. Clear Write Permission
6. Clear Execute Permission
7. Display Permissions
8. Exit
Enter your choice: 2

1. Set Read Permission
2. Set Write Permission
3. Set Execute Permission
4. Clear Read Permission
5. Clear Write Permission
6. Clear Execute Permission
7. Display Permissions
8. Exit
Enter your choice: 3

1. Set Read Permission
2. Set Write Permission
3. Set Execute Permission
4. Clear Read Permission
5. Clear Write Permission
6. Clear Execute Permission
7. Display Permissions
8. Exit
Enter your choice: 7
Permissions: R:1 W:1 X:1
```



```
1. Set Read Permission
2. Set Write Permission
3. Set Execute Permission
4. Clear Read Permission
5. Clear Write Permission
6. Clear Execute Permission
7. Display Permissions
8. Exit
Enter your choice: 5

1. Set Read Permission
2. Set Write Permission
3. Set Execute Permission
4. Clear Read Permission
5. Clear Write Permission
6. Clear Execute Permission
7. Display Permissions
8. Exit
Enter your choice: 7
Permissions: R:1 W:0 X:1

1. Set Read Permission
2. Set Write Permission
3. Set Execute Permission
4. Clear Read Permission
5. Clear Write Permission
6. Clear Execute Permission
7. Display Permissions
8. Exit
```

13.Problem 4: Network Packet Header

Problem Statement:

Write a C program to represent a network packet header using bit-fields. The program should:

Define a structure named PacketHeader with the following bit-fields:

version (4 bits): Protocol version (0-15).

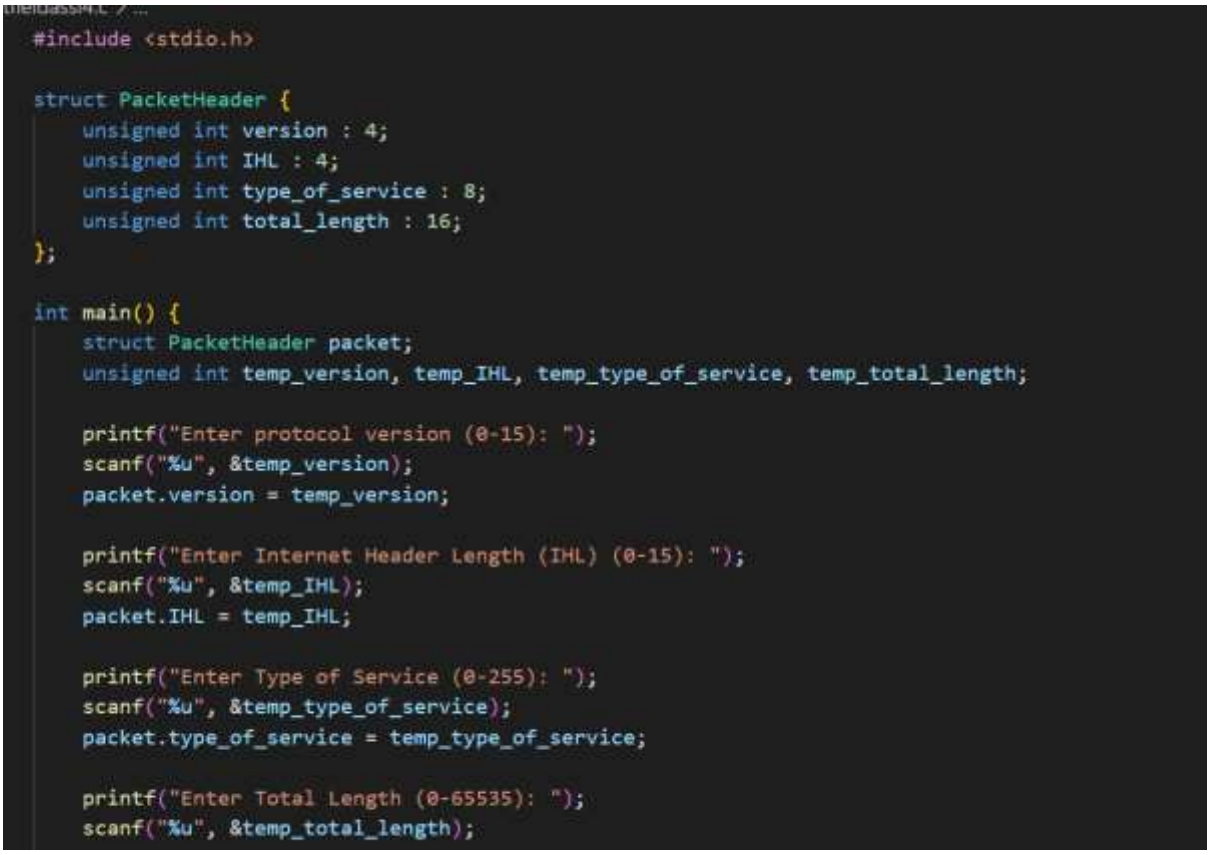
IHL (4 bits): Internet Header Length (0-15).

type_of_service (8 bits): Type of service.

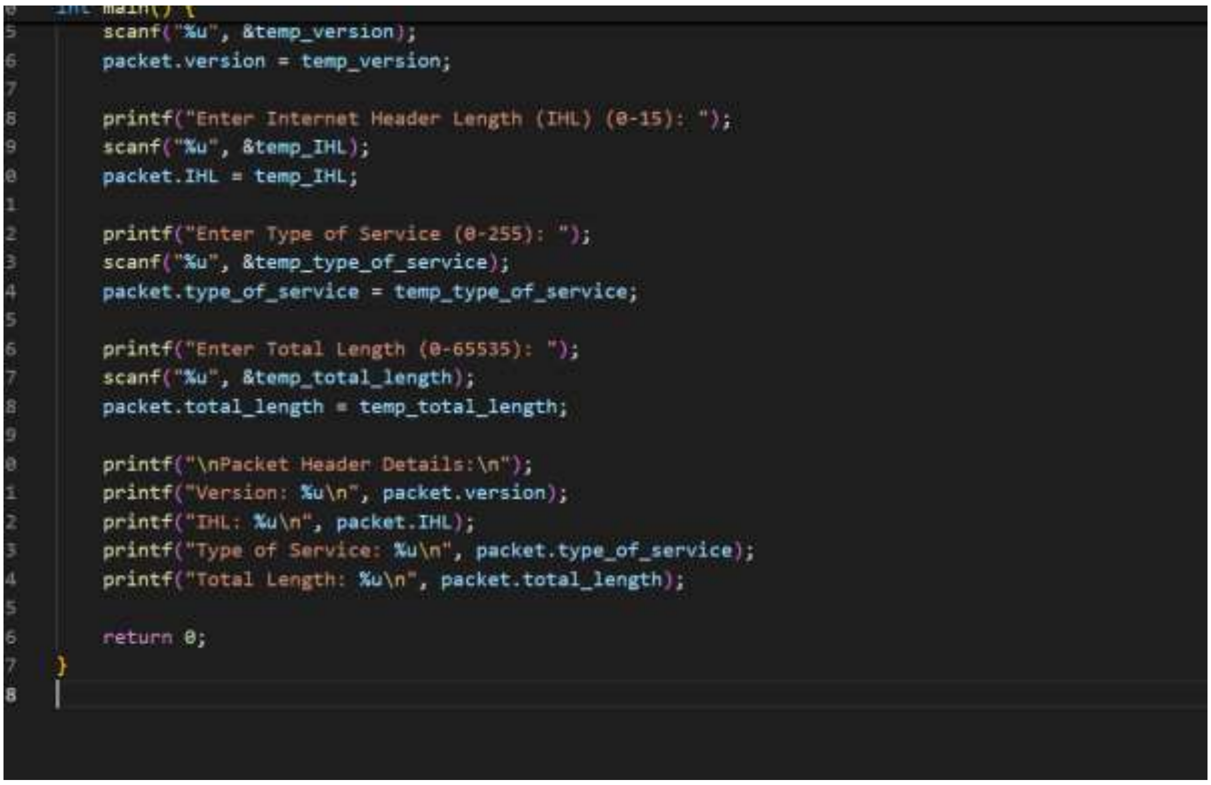
total_length (16 bits): Total packet length.

Allow the user to input values for each field and store them in the structure.

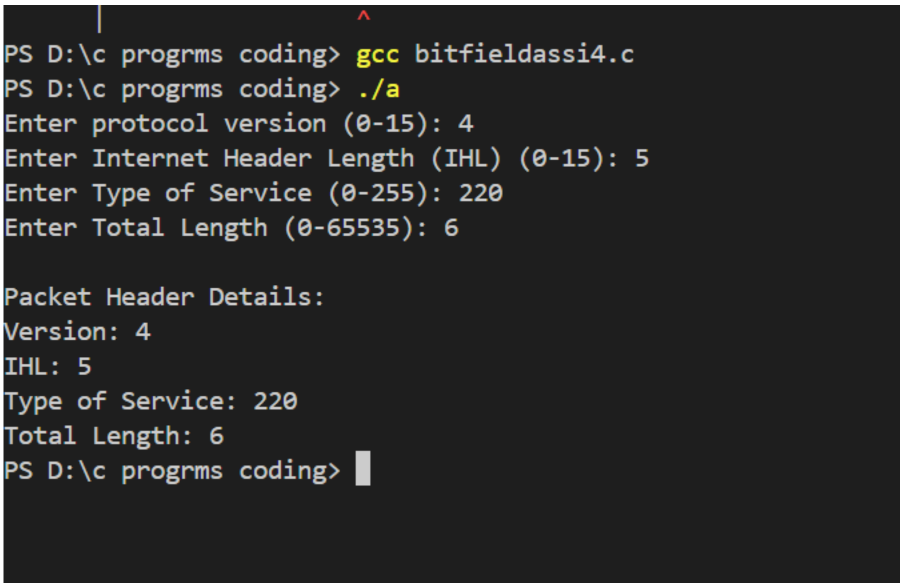
Display the packet header details in a structured format.



```c
#include <stdio.h>

struct PacketHeader {
    unsigned int version : 4;
    unsigned int IHL : 4;
    unsigned int type_of_service : 8;
    unsigned int total_length : 16;
};

int main() {
    struct PacketHeader packet;
    unsigned int temp_version, temp_IHL, temp_type_of_service, temp_total_length;

    printf("Enter protocol version (0-15): ");
    scanf("%u", &temp_version);
    packet.version = temp_version;

    printf("Enter Internet Header Length (IHL) (0-15): ");
    scanf("%u", &temp_IHL);
    packet.IHL = temp_IHL;

    printf("Enter Type of Service (0-255): ");
    scanf("%u", &temp_type_of_service);
    packet.type_of_service = temp_type_of_service;

    printf("Enter Total Length (0-65535): ");
    scanf("%u", &temp_total_length);
```



```c
int main() {
    scanf("%u", &temp_version);
    packet.version = temp_version;

    printf("Enter Internet Header Length (IHL) (0-15): ");
    scanf("%u", &temp_IHL);
    packet.IHL = temp_IHL;

    printf("Enter Type of Service (0-255): ");
    scanf("%u", &temp_type_of_service);
    packet.type_of_service = temp_type_of_service;

    printf("Enter Total Length (0-65535): ");
    scanf("%u", &temp_total_length);
    packet.total_length = temp_total_length;

    printf("\nPacket Header Details:\n");
    printf("Version: %u\n", packet.version);
    printf("IHL: %u\n", packet.IHL);
    printf("Type of Service: %u\n", packet.type_of_service);
    printf("Total Length: %u\n", packet.total_length);

    return 0;
}
```



```
      |                 ^
PS D:\c progrms coding> gcc bitfieldassi4.c
PS D:\c progrms coding> ./a
Enter protocol version (0-15): 4
Enter Internet Header Length (IHL) (0-15): 5
Enter Type of Service (0-255): 220
Enter Total Length (0-65535): 6

Packet Header Details:
Version: 4
IHL: 5
Type of Service: 220
Total Length: 6
PS D:\c progrms coding> 
```

14.Problem 5: Employee Work Hours Tracking

Problem Statement:

Write a C program to track employee work hours using bit-fields. The program should:

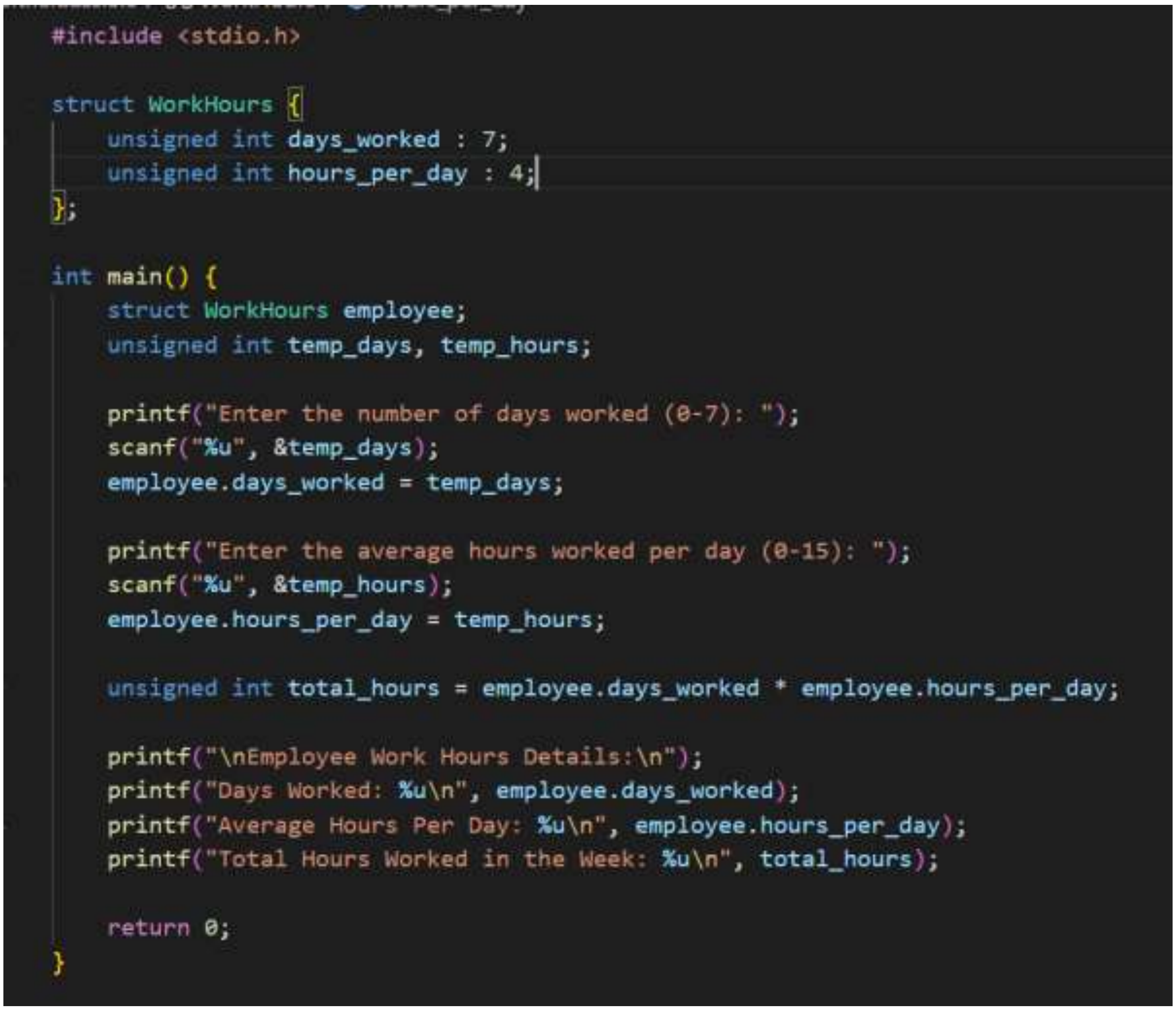
Define a structure named WorkHours with bit-fields:

days_worked (7 bits): Number of days worked in a week (0-7).

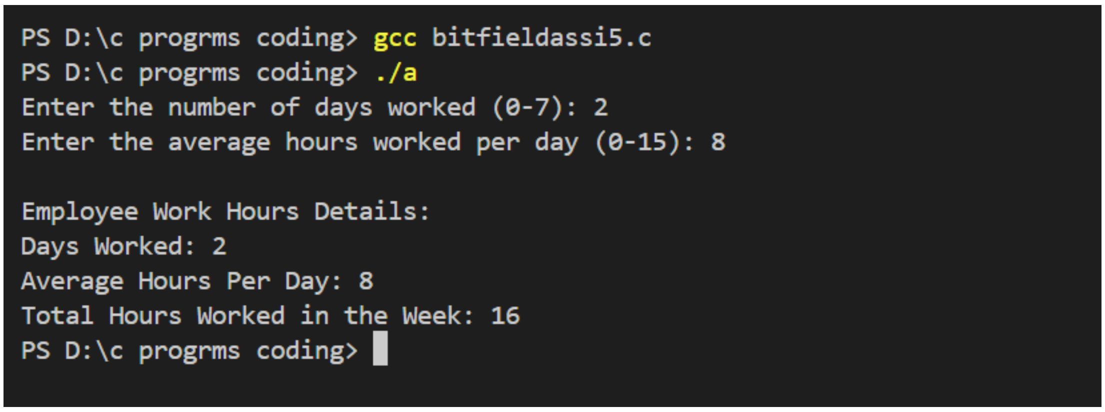
hours_per_day (4 bits): Average number of hours worked per day (0-15).

Allow the user to input the number of days worked and the average hours pe r day for an employee.

Calculate and display the total hours worked in the week.



```c
#include <stdio.h>

struct WorkHours {
    unsigned int days_worked : 7;
    unsigned int hours_per_day : 4;
};

int main() {
    struct WorkHours employee;
    unsigned int temp_days, temp_hours;

    printf("Enter the number of days worked (0-7): ");
    scanf("%u", &temp_days);
    employee.days_worked = temp_days;

    printf("Enter the average hours worked per day (0-15): ");
    scanf("%u", &temp_hours);
    employee.hours_per_day = temp_hours;

    unsigned int total_hours = employee.days_worked * employee.hours_per_day;

    printf("\nEmployee Work Hours Details:\n");
    printf("Days Worked: %u\n", employee.days_worked);
    printf("Average Hours Per Day: %u\n", employee.hours_per_day);
    printf("Total Hours Worked in the Week: %u\n", total_hours);

    return 0;
}
```



```
PS D:\c progrms coding> gcc bitfieldassi5.c
PS D:\c progrms coding> ./a
Enter the number of days worked (0-7): 2
Enter the average hours worked per day (0-15): 8

Employee Work Hours Details:
Days Worked: 2
Average Hours Per Day: 8
Total Hours Worked in the Week: 16
PS D:\c progrms coding> 
```