1.

```
/*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
 */

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

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

int main() {
    struct Employee *ptr;
    int n;

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

    if (n <= 0) {
        printf("Number of employees must be a positive integer.\n");
        return 1;
    }

    ptr = (struct Employee *)malloc(n * sizeof(struct Employee));
    if (ptr == NULL) {
        printf("Memory allocation failed for employees.\n");
        return 1;
    }

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

```
        int unique;
        do {
            unique = 1;
            printf("ID: ");
            scanf("%d", &(ptr + i)->id);

            for (int j = 0; j < i; j++) {
                if ((ptr + i)->id == (ptr + j)->id) {
                    printf("ID already exists. Please enter a unique ID.\n");
                    unique = 0;
                    break;
                }
            }
        } while (!unique);

        getchar();

        printf("Name: ");
        fgets((ptr + i)->name, sizeof((ptr + i)->name), stdin);
        (ptr + i)->name[strcspn((ptr + i)->name, "\n")] = 0;

        printf("Salary: ");
        scanf("%f", &(ptr + i)->salary);
    }

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

    int searchId, found = 0;
    printf("\nEnter ID to search for: ");
    scanf("%d", &searchId);

    printf("\nSearch Result:\n");
    for (int i = 0; i < n; i++) {
        if ((ptr + i)->id == searchId) {
            printf("ID: %d, Name: %s, Salary: %.2f\n",
                   (ptr + i)->id, (ptr + i)->name, (ptr + i)->salary);
            found = 1;
            break;
        }
```

```c
    }

    if (!found) {
        printf("Employee not found.\n");
    }

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

```
PS C:\Users\betti\Desktop\Training\Day14> ./task1
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

Employee Details:
ID: 101, Name: Alice, Salary: 50000.00
ID: 102, Name: Bob, Salary: 60000.00
ID: 103, Name: Charlie, Salary: 55000.00

Enter ID to search for: 102

Search Result:
ID: 102, Name: Bob, Salary: 60000.00
```

2.

```c
/*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.
 */

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

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

int main() {
    struct Book *bPtr;
    int n;

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

    if (n <= 0) {
        printf("Invalid number of books. Exiting program.\n");
        return 1;
    }

    bPtr = (struct Book *)malloc(n * sizeof(struct Book));
    if (bPtr == NULL) {
```

```
        printf("Memory allocation failed.\n");
        return 1;
    }

    // Input details for each book
    for (int i = 0; i < n; i++) {
        printf("\nEnter details for book %d:\n", i + 1);
        int unique;
        do {
            unique = 1;
            printf("ID: ");
            scanf("%d", &(bPtr[i].id));

            for (int j = 0; j < i; j++) {
                if (bPtr[i].id == bPtr[j].id) {
                    printf("ID already exists. Please enter a unique ID.\n");
                    unique = 0;
                    break;
                }
            }
        } while (!unique);

        getchar(); // Clear the newline character
        printf("Title: ");
        fgets(bPtr[i].title, sizeof(bPtr[i].title), stdin);
        bPtr[i].title[strcspn(bPtr[i].title, "\n")] = '\0'; // Remove newline

        do {
            printf("Price: ");
            scanf("%f", &bPtr[i].price);
            if (bPtr[i].price < 0) {
                printf("Price cannot be negative. Please enter a valid
price.\n");
            }
        } while (bPtr[i].price < 0);
    }

    // Menu for operations
    while (1) {
        int choice;
        printf("\n1. Display details\n");
        printf("2. Find cheapest book\n");
        printf("3. Update price\n");
```

```
        printf("4. Exit\n");
        printf("\nChoose an option: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("\nDetails of all books:\n");
                for (int i = 0; i < n; i++) {
                    printf("ID: %d\nTitle: %s\nPrice: %.2f\n\n", bPtr[i].id,
bPtr[i].title, bPtr[i].price);
                }
                break;

            case 2: {
                float minPrice = bPtr[0].price;
                int minIndex = 0;
                for (int i = 1; i < n; i++) {
                    if (bPtr[i].price < minPrice) {
                        minPrice = bPtr[i].price;
                        minIndex = i;
                    }
                }
                printf("\nCheapest Book Details:\n");
                printf("ID: %d\nTitle: %s\nPrice: %.2f\n\n", bPtr[minIndex].id,
bPtr[minIndex].title, bPtr[minIndex].price);
                break;
            }

            case 3: {
                int updateId, found = 0;
                printf("\nEnter the ID of the book to update price: ");
                scanf("%d", &updateId);
                for (int i = 0; i < n; i++) {
                    if (bPtr[i].id == updateId) {
                        printf("Enter new price: ");
                        scanf("%f", &bPtr[i].price);
                        printf("Price updated successfully.\n");
                        found = 1;
                        break;
                    }
                }
                if (!found) {
                    printf("Book with ID %d not found.\n", updateId);
```

```
                }
                break;
            }

            case 4:
                printf("\nExiting...\n");
                free(bPtr);
                return 0;

            default:
                printf("Invalid choice. Please try again.\n");
                break;
        }
    }

    return 0;
}
```

```
PS C:\Users\betti\Desktop\Training\Day14> ./task2
Enter the number of books: 2

Enter details for book 1:
ID: 101
Title: The Kite Runner
Price: 300

Enter details for book 2:
ID: 102
Title: It Ends With Us
Price: 250

1. Display details
2. Find cheapest book
3. Update price
4. Exit

Choose an option: 1

Details of all books:
ID: 101
Title: The Kite Runner
Price: 300.00

ID: 102
Title: It Ends With Us
Price: 250.00


1. Display details
2. Find cheapest book
3. Update price
4. Exit
```

```
Choose an option: 2

Cheapest Book Details:
ID: 102
Title: It Ends With Us
Price: 250.00


1. Display details
2. Find cheapest book
3. Update price
4. Exit

Choose an option: 3

Enter the ID of the book to update price: 101
Enter new price: 350
Price updated successfully.

1. Display details
2. Find cheapest book
3. Update price
4. Exit

Choose an option: 1

Details of all books:
ID: 101
Title: The Kite Runner
Price: 350.00

ID: 102
Title: It Ends With Us
Price: 250.00
```

3.

```
/*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).
```

```c
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.
 */

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

struct Point {
    float x;
    float y;
};

void displayPoints(struct Point *, int);
float findDistance(struct Point *, int, int, int);
void findClosestPair(struct Point *, int);

int main() {
    struct Point *ptr;
    int n;
    printf("Enter the number of points: ");
    scanf("%d", &n);
    if (n <= 0) {
        printf("Invalid number of points.\n");
        return 1;
    }

    ptr = (struct Point *)malloc(n * sizeof(struct Point));
    if (ptr == NULL) {
        printf("Memory allocation failed.\n");
        return 1;
    }

    for (int i = 0; i < n; i++) {
        printf("Enter the coordinates of point %d (x y): ", i + 1);
        scanf("%f %f", &ptr[i].x, &ptr[i].y);
    }
```

```c
    while (1) {
        printf("\n1. Display Points\n");
        printf("2. Find Distance\n");
        printf("3. Find Closest Pair\n");
        printf("4. Exit\n");
        printf("\nChoose an option: ");
        int choice;
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                displayPoints(ptr, n);
                break;
            case 2: {
                int index1, index2;
                printf("Enter the indices of the points to find the distance (1-
based indexing): ");
                scanf("%d %d", &index1, &index2);
                float dist = findDistance(ptr, n, index1 - 1, index2 - 1);
                if (dist >= 0) {
                    printf("Distance between points %d and %d: %.2f\n", index1,
index2, dist);
                }
                break;
            }
            case 3:
                findClosestPair(ptr, n);
                break;
            case 4:
                free(ptr);
                return 0;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}

void displayPoints(struct Point *pPtr, int noOfPoints) {
    printf("Coordinates of all points:\n");
    for (int i = 0; i < noOfPoints; i++) {
        printf("Point %d: (%.2f, %.2f)\n", i + 1, pPtr[i].x, pPtr[i].y);
```

```c
    }
}

float findDistance(struct Point *pPtr, int noOfPoints, int i1, int i2) {
    if (i1 < 0 || i1 >= noOfPoints || i2 < 0 || i2 >= noOfPoints) {
        printf("Invalid indices. Please enter indices between 1 and %d.\n",
noOfPoints);
        return -1;
    }
    float dx = pPtr[i1].x - pPtr[i2].x;
    float dy = pPtr[i1].y - pPtr[i2].y;
    return sqrt(dx * dx + dy * dy);
}

void findClosestPair(struct Point *pPtr, int noOfPoints) {
    if (noOfPoints < 2) {
        printf("Not enough points to find the closest pair.\n");
        return;
    }

    float minDistance = INFINITY;
    int index1 = 0, index2 = 1;

    for (int i = 0; i < noOfPoints; i++) {
        for (int j = i + 1; j < noOfPoints; j++) {
            float distance = sqrt(pow(pPtr[i].x - pPtr[j].x, 2) + pow(pPtr[i].y -
pPtr[j].y, 2));
            if (distance < minDistance) {
                minDistance = distance;
                index1 = i;
                index2 = j;
            }
        }
    }

    printf("Closest pair of points: (%.2f, %.2f) and (%.2f, %.2f)\n",
           pPtr[index1].x, pPtr[index1].y, pPtr[index2].x, pPtr[index2].y);
    printf("Distance between them: %.2f\n", minDistance);
}
```

```
PS C:\Users\betti\Desktop\Training\Day14> ./task3
Enter the number of points: 3
Enter the coordinates of point 1 (x y): 1 1
Enter the coordinates of point 2 (x y): 2 2
Enter the coordinates of point 3 (x y): 3 3

1. Display Points
2. Find Distance
3. Find Closest Pair
4. Exit

Choose an option: 1
Coordinates of all points:
Point 1: (1.00, 1.00)
Point 2: (2.00, 2.00)
Point 3: (3.00, 3.00)

1. Display Points
2. Find Distance
3. Find Closest Pair
4. Exit

Choose an option: 2
Enter the indices of the points to find the distance (1-based indexing): 1 2
Distance between points 1 and 2: 1.41


1. Display Points
2. Find Distance
3. Find Closest Pair
4. Exit

Choose an option: 3
Closest pair of points: (1.00, 1.00) and (2.00, 2.00)
Distance between them: 1.41
```

4.

```
/*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.



Sample Input/Output

Input:
Enter vehicle type (C for Car, B for Bike, S for Bus): C
Enter car model: Toyota Corolla
Output:
Vehicle Type: Car
Car Model: Toyota Corolla



Input:
Enter vehicle type (C for Car, B for Bike, S for Bus): B
Enter bike engine capacity (CC): 150
Output:
Vehicle Type: Bike
Engine Capacity: 150 CC

Input:
Enter vehicle type (C for Car, B for Bike, S for Bus): S
Enter number of seats in the bus: 50
Output:
Vehicle Type: Bus
Number of Seats: 50*/
```

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

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

struct VehicleInfo {
    char type;
    union Vehicle vehicle;
};

int main() {
    struct VehicleInfo vehicle;
    printf("Enter E to exit the system.\n");
    while(1) {
        printf("\nEnter vehicle type (C for Car, B for Bike, S for Bus): ");
        scanf(" %c", &vehicle.type);

        switch(vehicle.type)
        {
            case 'C':
                getchar();
                printf("Enter car model: ");
                scanf("%[^\n]", vehicle.vehicle.car_model);
                printf("\nVehicle Type: Car\n");
                printf("Car Model: %s\n", vehicle.vehicle.car_model);
                break;
            case 'B':
                printf("Enter bike engine capacity (CC): ");
                scanf("%d", &vehicle.vehicle.bike_cc);
                printf("\nVehicle Type: Bike\n");
                printf("Engine Capacity: %d CC\n", vehicle.vehicle.bike_cc);
                break;
            case 'S':
                printf("Enter number of seats in the bus: ");
                scanf("%d", &vehicle.vehicle.bus_seats);
                printf("\nVehicle Type: Bus\n");
                printf("Number of Seats: %d\n", vehicle.vehicle.bus_seats);
                break;
            case 'E':
```

```
                printf("\nExiting the system...\n");
                return 0;
            default:
                printf("Invalid vehicle type. Please enter C, B, or S.\n");
                continue;
        }
    }
    return 0;
}
```

```
PS C:\Users\betti\Desktop\Training\Day14> ./task4
Enter E to exit the system.

Enter vehicle type (C for Car, B for Bike, S for Bus): C
Enter car model: Toyota Corolla

Vehicle Type: Car
Car Model: Toyota Corolla

Enter vehicle type (C for Car, B for Bike, S for Bus): B
Enter bike engine capacity (CC): 150

Vehicle Type: Bike
Engine Capacity: 150 CC

Enter vehicle type (C for Car, B for Bike, S for Bus): S
Enter number of seats in the bus: 50

Vehicle Type: Bus
Number of Seats: 50

Enter vehicle type (C for Car, B for Bike, S for Bus): E

Exiting the system...
```

5.

```
/*Problem 1: Traffic Light System
```

```c
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"*/

#include <stdio.h>

enum TrafficLight {
    RED = 0,
    YELLOW = 1,
    GREEN = 2
};

int main() {
    enum TrafficLight currentLight; // Variable of type TrafficLight

    printf("Traffic Light System (press 3 to exit)\n");

    while (1) {
        printf("Enter current Traffic Light color (0 for RED, 1 for YELLOW, 2 for
GREEN): ");
        scanf("%d", &currentLight);

        if (currentLight == 3) {
            printf("Exiting the system...\n");
            break;
        }

        switch (currentLight) {
            case 0:
                printf("Stop\n");
                break;
            case 1:
                printf("Ready to move\n");
                break;
            case 2:
                printf("Go\n");
```

```
                break;
            default:
                printf("Invalid input. Please enter a valid choice of color.\n");
                break;
        }
    }

    return 0;
}
```

```
PS C:\Users\betti\Desktop\Training\Day14> ./task5
Traffic Light System (press 3 to exit)
Enter current Traffic Light color (0 for RED, 1 for YELLOW, 2 for GREEN): 0
Stop
Enter current Traffic Light color (0 for RED, 1 for YELLOW, 2 for GREEN): 1
Ready to move
Enter current Traffic Light color (0 for RED, 1 for YELLOW, 2 for GREEN): 2
Go
Enter current Traffic Light color (0 for RED, 1 for YELLOW, 2 for GREEN): 3
Exiting the system...
```

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
 */

#include <stdio.h>

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

int main() {
    enum Weekday day;
    printf("Enter a number (1 to 7) representing the day of the week: ");
    scanf("%d", &day);
```

```c
    switch (day) {
        case MONDAY:
            printf("Monday is a weekday.\n");
            break;
        case TUESDAY:
            printf("Tuesday is a weekday.\n");
            break;
        case WEDNESDAY:
            printf("Wednesday is a weekday.\n");
            break;
        case THURSDAY:
            printf("Thursday is a weekday.\n");
            break;
        case FRIDAY:
            printf("Friday is a weekday.\n");
            break;
        case SATURDAY:
            printf("Saturday is a weekend.\n");
            break;
        case SUNDAY:
            printf("Sunday is a weekend.\n");
            break;
        default:
            printf("Invalid day number! Please enter a number between 1 and
7.\n");
            return 1;
    }

    return 0;
}
```

```
PS C:\Users\betti\Desktop\Training\Day14> ./task6
Enter a number (1 to 7) representing the day of the week: 2
Tuesday is a weekday.
```

7.

```c
/*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:
```

```c
For CIRCLE: Radius
For RECTANGLE: Length and breadth
For TRIANGLE: Base and height
Calculate and display the area of the selected shape.
 */

#include <stdio.h>

enum Shape {
    CIRCLE,
    RECTANGLE,
    TRIANGLE
};

int main() {
    enum Shape shape;

    while(1) {
        printf("\nEnter the shape (0 for CIRCLE, 1 for RECTANGLE, 2 for
TRIANGLE): ");
        scanf("%d", &shape);

        double area;

        switch (shape) {
            case CIRCLE:
                double radius;
                printf("Enter the radius: ");
                scanf("%lf", &radius);
                area = 3.14 * radius * radius;
                printf("Area: %.2f\n", area);
                break;

            case RECTANGLE:
                double length, breadth;
                printf("Enter the length: ");
                scanf("%lf", &length);
                printf("Enter the breadth: ");
                scanf("%lf", &breadth);
                area = length * breadth;
                printf("Area: %.2f\n", area);
                break;
```

```
            case TRIANGLE:
                double base, height;
                printf("Enter the base: ");
                scanf("%lf", &base);
                printf("Enter the height: ");
                scanf("%lf", &height);
                area = 0.5 * base * height;
                printf("Area: %.2f\n", area);
                break;

            default:
                printf("Invalid shape selection.\n");
                return 1;
        }
    }
}
```

```
PS C:\Users\betti\Desktop\Training\Day14> ./task7

Enter the shape (0 for CIRCLE, 1 for RECTANGLE, 2 for TRIANGLE): 0
Enter the radius: 2
Area: 12.56

Enter the shape (0 for CIRCLE, 1 for RECTANGLE, 2 for TRIANGLE): 1
Enter the length: 2
Enter the breadth: 3
Area: 6.00

Enter the shape (0 for CIRCLE, 1 for RECTANGLE, 2 for TRIANGLE): 2
Enter the base: 3
Enter the height: 4
Area: 6.00
```

8.

```
/*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)
```

```c
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.
 */

#include <stdio.h>

enum ErrorCode {
    SUCCESS,
    FILE_NOT_FOUND,
    ACCESS_DENIED,
    OUT_OF_MEMORY,
    UNKNOWN_ERROR
};

enum ErrorCode simulateOperation();

int main() {
    enum ErrorCode errorCode;

    errorCode = simulateOperation();

    switch (errorCode) {
        case SUCCESS:
            printf("No error occurred. Operation was successful.\n");
            break;
        case FILE_NOT_FOUND:
            printf("Error: File not found.\n");
            break;
        case ACCESS_DENIED:
            printf("Error: Access denied.\n");
            break;
        case OUT_OF_MEMORY:
            printf("Error: Out of memory.\n");
            break;
        case UNKNOWN_ERROR:
            printf("Error: Unknown error occurred.\n");
            break;
        default:
            printf("Invalid error code.\n");
            break;
    }

    return 0;
```

```c
}

enum ErrorCode simulateOperation() {
    int choice;

    printf("Simulate an operation:\n");
    printf("0. Success\n");
    printf("1. File not found\n");
    printf("2. Access denied\n");
    printf("3. Out of memory\n");
    printf("4. Unknown error\n");
    printf("Enter the error code to simulate (0 to 4): ");
    scanf("%d", &choice);

    if (choice >= SUCCESS && choice <= UNKNOWN_ERROR) {
        return (enum ErrorCode)choice;
    } else {
        printf("Invalid choice. Returning UNKNOWN_ERROR by default.\n");
        return UNKNOWN_ERROR;
    }
}
```

```
PS C:\Users\betti\Desktop\Training\Day14> ./task8
Simulate an operation:
0. Success
1. File not found
2. Access denied
3. Out of memory
4. Unknown error
Enter the error code to simulate (0 to 4): 0
No error occurred. Operation was successful.
PS C:\Users\betti\Desktop\Training\Day14> ./task8
Simulate an operation:
0. Success
1. File not found
2. Access denied
3. Out of memory
4. Unknown error
Enter the error code to simulate (0 to 4): 3
Error: Out of memory.
```

9.

```c
/*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."*/

#include <stdio.h>

enum UserRole { ADMIN, EDITOR, VIEWER, GUEST };
```

```c
int main() {
    enum UserRole role;
    while(1) {
        printf("\nEnter user role (0 for ADMIN, 1 for EDITOR, 2 for VIEWER, 3 for
GUEST): ");
        scanf("%d", &role);
        printf("Permissions for user role %d:\n", role);
        switch (role) {
            case ADMIN:
                printf("Full access to the system.\n");
                break;
            case EDITOR:
                printf("Can edit content but not manage users.\n");
                break;
            case VIEWER:
                printf("Can view content only.\n");
                break;
            case GUEST:
                printf("Limited access, view public content only.\n");
                break;
            default:
                printf("Invalid role entered. Please try again.\n");
                return 1;
        }
    }
    return 0;
}
```

```
PS C:\Users\betti\Desktop\Training\Day14> ./task9

Enter user role (0 for ADMIN, 1 for EDITOR, 2 for VIEWER, 3 for GUEST): 0
Permissions for user role 0:
Full access to the system.

Enter user role (0 for ADMIN, 1 for EDITOR, 2 for VIEWER, 3 for GUEST): 1
Permissions for user role 1:
Can edit content but not manage users.

Enter user role (0 for ADMIN, 1 for EDITOR, 2 for VIEWER, 3 for GUEST): 2
Permissions for user role 2:
Can view content only.

Enter user role (0 for ADMIN, 1 for EDITOR, 2 for VIEWER, 3 for GUEST): 3
Permissions for user role 3:
Limited access, view public content only.

Enter user role (0 for ADMIN, 1 for EDITOR, 2 for VIEWER, 3 for GUEST): 4
Permissions for user role 4:
Invalid role entered. Please try again.
```

10.

```c
/*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.*/

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

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

```c
};

int main() {
    struct Date *datePtr;
    datePtr = malloc(5 * sizeof(struct Date));
    if(datePtr == NULL) {
        printf("Memory allocation failed!\n");
        return 1;
    }

    for (int i = 0; i < 5; i++) {
        int day, month, year;
        printf("Enter date %d (DD MM YYYY): ", i + 1);
        scanf("%d %d %d", &day, &month, &year);

        (datePtr + i)->day = day;
        (datePtr + i)->month = month;
        (datePtr + i)->year = year;
    }

    printf("\nStored Dates:\n");
    for(int i = 0; i < 5; i++) {
        printf("%02d-%02d-%04d\n", (datePtr+i)->day, (datePtr+i)->month,
(datePtr+i)->year);
    }

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

```
PS C:\Users\betti\Desktop\Training\Day14> ./task10
Enter date 1 (DD MM YYYY): 12 12 2024
Enter date 2 (DD MM YYYY): 17 12 2024
Enter date 3 (DD MM YYYY): 07 12 2024
Enter date 4 (DD MM YYYY): 25 12 2024
Enter date 5 (DD MM YYYY): 05 12 2024

Stored Dates:
12-12-2024
17-12-2024
07-12-2024
25-12-2024
05-12-2024
```

11.

```c
/*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).*/

#include <stdio.h>

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

int main() {
```

```
    struct DeviceStatus deviceStatus;
    deviceStatus.power = 0;
    deviceStatus.connection = 0;
    deviceStatus.error = 0;

    char input;
    while (1) {
        int power, connection, error;
        printf("\nCurrent Device Status:\n");
        printf("Power: %s\n", deviceStatus.power? "ON" : "OFF");
        printf("Connection: %s\n", deviceStatus.connection? "CONNECTED" :
"DISCONNECTED");
        printf("Error: %s\n", deviceStatus.error? "YES" : "NO");
        printf("\nEnter 1 to set and 0 to reset:\n");
        printf("Power: ");
        scanf("%d", &power);
        printf("Connection: ");
        scanf("%d", &connection);
        printf("Error: ");
        scanf("%d", &error);

        deviceStatus.power = power;
        deviceStatus.connection = connection;
        deviceStatus.error = error;

        printf("\nUpdated Device Status:\n");
        printf("Power: %s\n", deviceStatus.power? "ON" : "OFF");
        printf("Connection: %s\n", deviceStatus.connection? "CONNECTED" :
"DISCONNECTED");
        printf("Error: %s\n", deviceStatus.error? "YES" : "NO");

        printf("\nDo you want to continue? (Y/N): ");
        scanf(" %c", &input);
        if (input == 'N' || input == 'n')
        {
            break;
        }

        printf("\n");
    }

        return 0;
}
```

```
Current Device Status:
Power: OFF
Connection: DISCONNECTED
Error: NO

Enter 1 to set and 0 to reset:
Power: 1
Connection: 0
Error: 1

Updated Device Status:
Power: ON
Connection: DISCONNECTED
Error: YES

Do you want to continue? (Y/N): y


Current Device Status:
Power: ON
Connection: DISCONNECTED
Error: YES

Enter 1 to set and 0 to reset:
Power: 1
Connection: 1
Error: 0

Updated Device Status:
Power: ON
Connection: CONNECTED
Error: NO

Do you want to continue? (Y/N): n
```

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.
```

```c
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 read : 1;
    unsigned write : 1;
    unsigned execute : 1;
};

int main() {
    struct FilePermissions permissions;
    permissions.read = 0;
    permissions.write = 0;
    permissions.execute = 0;

    printf("\nCurrent permissions: R:%d W:%d X:%d\n", permissions.read,
permissions.write, permissions.execute);

    int read, write, execute;
    printf("\nEnter 1 to grant and 0 to deny permissions:\n");
    printf("Read: ");
    scanf("%d", &read);
    printf("Write: ");
    scanf("%d", &write);
    printf("Execute: ");
    scanf("%d", &execute);

    permissions.read = read;
    permissions.write = write;
    permissions.execute = execute;

    printf("\nUpdated permissions: R:%d W:%d X:%d\n", permissions.read,
permissions.write, permissions.execute);

    return 0;
}
```

```
PS C:\Users\betti\Desktop\Training\Day14> ./task12

Current permissions: R:0 W:0 X:0

Enter 1 to grant and 0 to deny permissions:
Read: 1
Write: 0
Execute: 0

Updated permissions: R:1 W:0 X:0
```

13.

```c
/*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.
*/

#include <stdio.h>

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

int main() {
    struct PacketHeader packet;
    int version, IHL, type_of_service, total_length;
```

```c
    printf("Enter the protocol version (0-15): ");
    scanf("%d", &version);
    printf("Enter the Internet Header Length (0-15): ");
    scanf("%d", &IHL);
    printf("Enter the Type of Service: ");
    scanf("%d", &type_of_service);
    printf("Enter the Total Packet Length: ");
    scanf("%d", &total_length);

    packet.version = version;
    packet.IHL = IHL;
    packet.type_of_service = type_of_service;
    packet.total_length = total_length;

    printf("\n-----------------------------------\n");
    printf("| Field               | Value      |\n");
    printf("-----------------------------------\n");
    printf("| Protocol Version    | %-10d |\n", packet.version);
    printf("| Header Length (IHL) | %-10d |\n", packet.IHL);
    printf("| Type of Service     | %-10d |\n", packet.type_of_service);
    printf("| Total Length        | %-10d |\n", packet.total_length);
    printf("-----------------------------------\n");

    return 0;
}
```

```
PS C:\Users\betti\Desktop\Training\Day14> ./task13
Enter the protocol version (0-15): 15
Enter the Internet Header Length (0-15): 12
Enter the Type of Service: 20
Enter the Total Packet Length: 1084

-----------------------------------
| Field               | Value      |
-----------------------------------
| Protocol Version    | 15         |
| Header Length (IHL) | 12         |
| Type of Service     | 20         |
| Total Length        | 1084       |
-----------------------------------
```

14.

```c
/*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 per day
for an employee.
Calculate and display the total hours worked in the week.*/

#include <stdio.h>

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

int main() {
    struct WorkHours employee_work_hours;
    int days_worked;
    int hours_per_day;

    printf("Enter the number of days worked in a week (0-7): ");
    scanf("%d", &days_worked);
    printf("Enter the average hours worked per day (0-15): ");
    scanf("%d", &hours_per_day);

    employee_work_hours.days_worked = days_worked;
    employee_work_hours.hours_per_day = hours_per_day;

    printf("\nEmployee Work Hours Summary:\n");
    printf("Days Worked: %d\n", employee_work_hours.days_worked);
    printf("Average Hours per Day: %d\n", employee_work_hours.hours_per_day);
    printf("Total Hours Worked in the Week: %d\n",
employee_work_hours.days_worked * employee_work_hours.hours_per_day);

    return 0;
}
```

```
PS C:\Users\betti\Desktop\Training\Day14> ./task14
Enter the number of days worked in a week (0-7): 2
Enter the average hours worked per day (0-15): 6

Employee Work Hours Summary:
Days Worked: 2
Average Hours per Day: 6
Total Hours Worked in the Week: 12
```