After assesment homework:

Problem 1: Patient Information Management System

Description: Create a menu-driven program to manage patient information, including basic details, medical history, and current medications.

Menu Options:

1. Add New Patient
2. View Patient Details
3. Update Patient Information
4. Delete Patient Record
5. List All Patients
6. Exit

Requirements:

7. Use variables to store patient details.
8. Utilize static and const for immutable data such as hospital name.
9. Implement switch case for menu selection.
10. Employ loops for iterative tasks like listing patients.
11. Use pointers for dynamic memory allocation.
12. Implement functions for CRUD operations.
13. Utilize arrays for storing multiple patient records.
14. Use structures for organizing patient data.
15. Apply nested structures for detailed medical history.
16. Use unions for optional data fields.
17. Employ nested unions for multi-type data entries.

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


#define HOSPITAL_NAME "City Hospital"


typedef struct {

   char name[50];
```

```
    int age;
    char medications[100];
} Patient;

Patient patients[100];
int patientCount = 0;

void addPatient() {
    Patient newPatient;
    printf("Enter patient name: ");
    scanf("%s", newPatient.name);
    printf("Enter age: ");
    scanf("%d", &newPatient.age);
    printf("Enter medications: ");
    scanf("%s", newPatient.medications);
    patients[patientCount++] = newPatient;
    printf("Patient added successfully!\n");
}

void listPatients() {
    printf("\nList of all patients:\n");
    for (int i = 0; i < patientCount; ++i) {
        printf("- %s, Age: %d, Medications: %s\n", patients[i].name, patients[i].age,
patients[i].medications);
    }
}

int main() {
```

```
    int choice;
    do {
        printf("\n%s Patient Management System\n", HOSPITAL_NAME);
        printf("1. Add New Patient\n");
        printf("2. List All Patients\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1: addPatient(); break;
            case 2: listPatients(); break;
            case 3: printf("Exiting...\n"); break;
            default: printf("Invalid choice. Try again.\n");
        }
    } while (choice != 3);

    return 0;
}
```

Problem 2: Hospital Inventory Management

Description: Design a system to manage the inventory of medical supplies.

Menu Options:

1. Add Inventory Item
2. View Inventory Item
3. Update Inventory Item
4. Delete Inventory Item
5. List All Inventory Items
6. Exit

Requirements:

7. Declare variables for inventory details.
8. Use static and const for fixed supply details.
9. Implement switch case for different operations like adding, deleting, and viewing inventory.
10. Utilize loops for repetitive inventory checks.
11. Use pointers to handle inventory records.
12. Create functions for managing inventory.
13. Use arrays to store inventory items.
14. Define structures for each supply item.
15. Use nested structures for detailed item specifications.
16. Employ unions for variable item attributes.
17. Implement nested unions for complex item data types.

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


#define HOSPITAL_NAME "City Hospital"


typedef struct {

   char manufacturer[50];

   char expiryDate[15];

} ItemDetails;


union ItemAttributes {

   int quantity;

   float weight;

};


typedef struct {

   char name[50];

   ItemDetails details;

   union ItemAttributes attributes;
```

```
    int isQuantity;
} InventoryItem;

InventoryItem inventory[100];
int itemCount = 0;

void addInventoryItem() {
    InventoryItem newItem;
    printf("Enter item name: ");
    scanf("%s", newItem.name);
    printf("Enter manufacturer: ");
    scanf("%s", newItem.details.manufacturer);
    printf("Enter expiry date: ");
    scanf("%s", newItem.details.expiryDate);
    printf("Enter 1 for quantity, 2 for weight: ");
    int choice;
    scanf("%d", &choice);
    if (choice == 1) {
        printf("Enter quantity: ");
        scanf("%d", &newItem.attributes.quantity);
        newItem.isQuantity = 1;
    } else {
        printf("Enter weight: ");
        scanf("%f", &newItem.attributes.weight);
        newItem.isQuantity = 0;
    }
    inventory[itemCount++] = newItem;
    printf("Inventory item added successfully!\n");
```

```
}

void listInventoryItems() {
    printf("\nList of all inventory items:\n");
    for (int i = 0; i < itemCount; ++i) {
        printf("- %s, Manufacturer: %s, Expiry Date: %s, ", inventory[i].name,
inventory[i].details.manufacturer, inventory[i].details.expiryDate);
        if (inventory[i].isQuantity) {
            printf("Quantity: %d\n", inventory[i].attributes.quantity);
        } else {
            printf("Weight: %.2f kg\n", inventory[i].attributes.weight);
        }
    }
}

int main() {
    int choice;
    do {
        printf("\n%s Inventory Management System\n", HOSPITAL_NAME);
        printf("1. Add Inventory Item\n");
        printf("2. List All Inventory Items\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1: addInventoryItem(); break;
            case 2: listInventoryItems(); break;
```

```
            case 3: printf("Exiting...\n"); break;
            default: printf("Invalid choice. Try again.\n");
        }
    } while (choice != 3);

    return 0;
}
```

Problem 3: Medical Appointment Scheduling System

Description: Develop a system to manage patient appointments.

Menu Options:

1. Schedule Appointment
2. View Appointment
3. Update Appointment
4. Cancel Appointment
5. List All Appointments
6. Exit

Requirements:

7. Use variables for appointment details.
8. Apply static and const for non-changing data like clinic hours.
9. Implement switch case for appointment operations.
10. Utilize loops for scheduling.
11. Use pointers for dynamic data manipulation.
12. Create functions for appointment handling.
13. Use arrays for storing appointments.
14. Define structures for appointment details.
15. Employ nested structures for detailed doctor and patient information.
16. Utilize unions for optional appointment data.
17. Apply nested unions for complex appointment data.

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

```
#define CLINIC_HOURS "9 AM - 5 PM"

typedef struct {
    char name[50];
    int age;
    char contactNumber[15];
} PatientInfo;

typedef struct {
    char name[50];
    char specialization[50];
} DoctorInfo;

union OptionalData {
    char additionalNotes[100];
    int followUpDays;
};

typedef struct {
    PatientInfo patient;
    DoctorInfo doctor;
    char appointmentDate[15];
    char appointmentTime[10];
    union OptionalData optional;
```

```
    int hasNotes;
} Appointment;

Appointment appointments[100];
int appointmentCount = 0;

void scheduleAppointment() {
    Appointment newAppointment;
    printf("Enter patient name: ");
    scanf("%s", newAppointment.patient.name);
    printf("Enter patient age: ");
    scanf("%d", &newAppointment.patient.age);
    printf("Enter patient contact number: ");
    scanf("%s", newAppointment.patient.contactNumber);
    printf("Enter doctor name: ");
    scanf("%s", newAppointment.doctor.name);
    printf("Enter doctor specialization: ");
    scanf("%s", newAppointment.doctor.specialization);
    printf("Enter appointment date: ");
    scanf("%s", newAppointment.appointmentDate);
    printf("Enter appointment time: ");
    scanf("%s", newAppointment.appointmentTime);
    printf("Enter 1 for notes, 2 for follow-up days: ");
    int choice;
    scanf("%d", &choice);
```

```
    if (choice == 1) {
        printf("Enter additional notes: ");
        scanf("%s", newAppointment.optional.additionalNotes);
        newAppointment.hasNotes = 1;
    } else {
        printf("Enter follow-up days: ");
        scanf("%d", &newAppointment.optional.followUpDays);
        newAppointment.hasNotes = 0;
    }
    appointments[appointmentCount++] = newAppointment;
    printf("Appointment scheduled successfully!\n");
}

void listAppointments() {
    printf("\nList of all appointments:\n");
    for (int i = 0; i < appointmentCount; ++i) {
        printf("Patient: %s, Age: %d, Contact: %s\n",
appointments[i].patient.name, appointments[i].patient.age,
appointments[i].patient.contactNumber);
        printf("Doctor: %s, Specialization: %s\n", appointments[i].doctor.name,
appointments[i].doctor.specialization);
        printf("Date: %s, Time: %s\n", appointments[i].appointmentDate,
appointments[i].appointmentTime);
        if (appointments[i].hasNotes) {
            printf("Notes: %s\n", appointments[i].optional.additionalNotes);
        } else {
            printf("Follow-up Days: %d\n",
appointments[i].optional.followUpDays);
```

```
        }
    }
}

int main() {
    int choice;
    do {
        printf("\nMedical Appointment Scheduling System\n");
        printf("Clinic Hours: %s\n", CLINIC_HOURS);
        printf("1. Schedule Appointment\n");
        printf("2. List All Appointments\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1: scheduleAppointment(); break;
            case 2: listAppointments(); break;
            case 3: printf("Exiting...\n"); break;
            default: printf("Invalid choice. Try again.\n");
        }
    } while (choice != 3);

    return 0;
}
```

Problem 4: Patient Billing System

Description: Create a billing system for patients.

Menu Options:

1. Generate Bill
2. View Bill
3. Update Bill
4. Delete Bill
5. List All Bills
6. Exit

Requirements:

7. Declare variables for billing information.
8. Use static and const for fixed billing rates.
9. Implement switch case for billing operations.
10. Utilize loops for generating bills.
11. Use pointers for bill calculations.
12. Create functions for billing processes.
13. Use arrays for storing billing records.
14. Define structures for billing components.
15. Employ nested structures for detailed billing breakdown.
16. Use unions for variable billing elements.
17. Apply nested unions for complex billing scenarios.

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>



#define FIXED_CONSULTATION_FEE 100.00



typedef struct {

    char name[50];

    int age;

    char contactNumber[15];

} PatientInfo;
```

```
typedef struct {
    float consultationFee;
    float medicationCharges;
    float miscellaneousCharges;
} BillingDetails;

union Discount {
    float percentage;
    float fixedAmount;
};

typedef struct {
    PatientInfo patient;
    BillingDetails bill;
    union Discount discount;
    int isPercentage;
} Bill;

Bill bills[100];
int billCount = 0;

void generateBill() {
    Bill newBill;
    printf("Enter patient name: ");
    scanf("%s", newBill.patient.name);
```

```
    printf("Enter patient age: ");
    scanf("%d", &newBill.patient.age);
    printf("Enter patient contact number: ");
    scanf("%s", newBill.patient.contactNumber);
    newBill.bill.consultationFee = FIXED_CONSULTATION_FEE;
    printf("Enter medication charges: ");
    scanf("%f", &newBill.bill.medicationCharges);
    printf("Enter miscellaneous charges: ");
    scanf("%f", &newBill.bill.miscellaneousCharges);
    printf("Enter 1 for percentage discount, 2 for fixed discount amount: ");
    int choice;
    scanf("%d", &choice);
    if (choice == 1) {
        printf("Enter discount percentage: ");
        scanf("%f", &newBill.discount.percentage);
        newBill.isPercentage = 1;
    } else {
        printf("Enter fixed discount amount: ");
        scanf("%f", &newBill.discount.fixedAmount);
        newBill.isPercentage = 0;
    }
    bills[billCount++] = newBill;
    printf("Bill generated successfully!\n");
}
```

```
void listBills() {

    printf("\nList of all bills:\n");

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

        printf("Patient: %s, Age: %d, Contact: %s\n", bills[i].patient.name,
bills[i].patient.age, bills[i].patient.contactNumber);

        printf("Consultation Fee: %.2f, Medication Charges: %.2f,
Miscellaneous: %.2f\n",

               bills[i].bill.consultationFee, bills[i].bill.medicationCharges,
bills[i].bill.miscellaneousCharges);

        if (bills[i].isPercentage) {

            printf("Discount: %.2f%%\n", bills[i].discount.percentage);

        } else {

            printf("Discount: %.2f\n", bills[i].discount.fixedAmount);

        }

    }

}



int main() {

    int choice;

    do {

        printf("\nPatient Billing System\n");

        printf("1. Generate Bill\n");

        printf("2. List All Bills\n");

        printf("3. Exit\n");

        printf("Enter your choice: ");

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

```
        switch (choice) {
            case 1: generateBill(); break;
            case 2: listBills(); break;
            case 3: printf("Exiting...\n"); break;
            default: printf("Invalid choice. Try again.\n");
        }
    } while (choice != 3);

    return 0;
}
```

Problem 5: Medical Test Result Management

Description: Develop a system to manage and store patient test results

Menu Options:

1. Add Test Result
2. View Test Result
3. Update Test Result
4. Delete Test Result
5. List All Test Results
6. Exit

Requirements:

7. Declare variables for test results.
8. Use static and const for standard test ranges.
9. Implement switch case for result operations.
10. Utilize loops for result input and output.
11. Use pointers for handling result data.
12. Create functions for result management.
13. Use arrays for storing test results.
14. Define structures for test result details.
15. Employ nested structures for detailed test parameters.
16. Utilize unions for optional test data.
17. Apply nested unions for complex test result data.

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

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

#define NORMAL_HEMOGLOBIN_MIN 13.5
#define NORMAL_HEMOGLOBIN_MAX 17.5

typedef struct {
    char name[50];
    int age;
    char contactNumber[15];
} PatientInfo;

typedef struct {
    float hemoglobinLevel;
    float cholesterolLevel;
    char testDate[15];
} TestParameters;

union OptionalData {
    char comments[100];
    float additionalCharges;
};

typedef struct {
    PatientInfo patient;
```

```
    TestParameters test;
    union OptionalData optional;
    int hasComments;
} TestResult;

TestResult results[100];
int resultCount = 0;

void addTestResult() {
    TestResult newResult;
    printf("Enter patient name: ");
    scanf("%s", newResult.patient.name);
    printf("Enter patient age: ");
    scanf("%d", &newResult.patient.age);
    printf("Enter patient contact number: ");
    scanf("%s", newResult.patient.contactNumber);
    printf("Enter hemoglobin level: ");
    scanf("%f", &newResult.test.hemoglobinLevel);
    printf("Enter cholesterol level: ");
    scanf("%f", &newResult.test.cholesterolLevel);
    printf("Enter test date: ");
    scanf("%s", newResult.test.testDate);
    printf("Enter 1 for comments, 2 for additional charges: ");
    int choice;
    scanf("%d", &choice);
```

```
    if (choice == 1) {
        printf("Enter comments: ");
        scanf("%s", newResult.optional.comments);
        newResult.hasComments = 1;
    } else {
        printf("Enter additional charges: ");
        scanf("%f", &newResult.optional.additionalCharges);
        newResult.hasComments = 0;
    }
    results[resultCount++] = newResult;
    printf("Test result added successfully!\n");
}

void listTestResults() {
    printf("\nList of all test results:\n");
    for (int i = 0; i < resultCount; ++i) {
        printf("Patient: %s, Age: %d, Contact: %s\n", results[i].patient.name,
results[i].patient.age, results[i].patient.contactNumber);
        printf("Hemoglobin Level: %.2f, Cholesterol Level: %.2f, Test Date:
%s\n",
              results[i].test.hemoglobinLevel, results[i].test.cholesterolLevel,
results[i].test.testDate);
        if (results[i].hasComments) {
            printf("Comments: %s\n", results[i].optional.comments);
        } else {
            printf("Additional Charges: %.2f\n",
results[i].optional.additionalCharges);
```

```
        }
    }
}

int main() {
    int choice;
    do {
        printf("\nMedical Test Result Management\n");
        printf("1. Add Test Result\n");
        printf("2. List All Test Results\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1: addTestResult(); break;
            case 2: listTestResults(); break;
            case 3: printf("Exiting...\n"); break;
            default: printf("Invalid choice. Try again.\n");
        }
    } while (choice != 3);

    return 0;
}
```

Problem 1: Patient Queue Management
Description: Implement a linked list to manage a queue of patients waiting for consultation. Operations:
Create a new patient queue.
Insert a patient into the queue.
Display the current queue of patients.

Problem 2: Hospital Ward Allocation
Description: Use a linked list to allocate beds in a hospital ward. Operations:
Create a list of available beds.
Insert a patient into an available bed.
Display the current bed allocation.

Problem 3: Medical Inventory Tracking
Description: Maintain a linked list to track inventory items in a medical store. Operations:
Create an inventory list.
Insert a new inventory item.
Display the current inventory.

Problem 4: Doctor Appointment Scheduling
Description: Develop a linked list to schedule doctor appointments. Operations:
Create an appointment list.
Insert a new appointment.
Display all scheduled appointments.
Problem 5: Emergency Contact List
Description: Implement a linked list to manage emergency contacts for hospital staff. Operations:
Create a contact list.
Insert a new contact.
Display all emergency contacts.

Problem 6: Surgery Scheduling System
Description: Use a linked list to manage surgery schedules. Operations:
Create a surgery schedule.
Insert a new surgery into the schedule.
Display all scheduled surgeries.

Problem 7: Patient History Record
Description: Maintain a linked list to keep track of patient history records. Operations:
Create a history record list.
Insert a new record.
Display all patient history records.

Problem 8: Medical Test Tracking
Description: Implement a linked list to track medical tests for patients. Operations:
Create a list of medical tests.
Insert a new test result.
Display all test results.

Problem 9: Prescription Management System
Description: Use a linked list to manage patient prescriptions. Operations:
Create a prescription list.
Insert a new prescription.
Display all prescriptions.

Problem 10: Hospital Staff Roster
Description: Develop a linked list to manage the hospital staff roster. Operations:
Create a staff roster.
Insert a new staff member into the roster.
Display the current staff roster.

```
// 1.
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct PatientNode
{
    char name[50];
    struct PatientNode *next;
} *first = NULL;

// Function prototypes
void createPatientQueue(char names[][50], int n);
void displayPatientQueue(struct PatientNode *p);
void insertPatient(struct PatientNode *p, char name[]);

int main()
{
    char patientNames[][50] = {"Nanditha M", "Niharika C L", "Shama M G"};
    createPatientQueue(patientNames, 3);
    printf("Initial patient queue:\n");
    displayPatientQueue(first);
    printf("\nAdding a new patient to the queue:\n");
    insertPatient(first, "Ram");
    displayPatientQueue(first);
    return 0;
}

void createPatientQueue(char names[][50], int n)
{
    int i;
    struct PatientNode *temp, *last;
    first = (struct PatientNode *)malloc(sizeof(struct PatientNode));
    strcpy(first->name, names[0]);
    first->next = NULL;
    last = first;
```

```
    for (i = 1; i < n; i++)
    {
        temp = (struct PatientNode *)malloc(sizeof(struct PatientNode));
        strcpy(temp->name, names[i]);
        temp->next = NULL;
        last->next = temp;
        last = temp;
    }
}

void displayPatientQueue(struct PatientNode *p)
{
    while (p != NULL)
    {
        printf("Name: %s\n", p->name);
        p = p->next;
    }
}

void insertPatient(struct PatientNode *p, char name[])
{
    struct PatientNode *temp, *last = p;
    temp = (struct PatientNode *)malloc(sizeof(struct PatientNode));
    strcpy(temp->name, name);
    temp->next = NULL;
    while (last->next != NULL)
        last = last->next;
    last->next = temp;
}*/



//2.
*#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Define a structure for the bed
struct BedNode
{
    int bedNumber;
    char patientName[50];
    struct BedNode *next;
} *first = NULL, *last = NULL;

// Function Prototypes
void createNode(int bedCount);
void displayBedAllocation(struct BedNode *p);
void allocateBed(struct BedNode *p, int bedNumber, char patientName[]);
```

```
int main()
{
    int bedCount = 5;
    createNode(bedCount);


    printf("Initial Bed Allocation:\n");
    displayBedAllocation(first);


    printf("\nAllocating bed 2 to patient 'John Smith'\n");
    allocateBed(first, 2, "John Smith");

    printf("\nUpdated Bed Allocation:\n");
    displayBedAllocation(first);

    return 0;
}

void createNode(int bedCount)
{
    int i;
    struct BedNode *temp;


    first = (struct BedNode *)malloc(sizeof(struct BedNode));
    first->bedNumber = 1;
    strcpy(first->patientName, "Available");
    first->next = NULL;
    last = first;


    for (i = 2; i <= bedCount; i++)
    {
        temp = (struct BedNode *)malloc(sizeof(struct BedNode));
        temp->bedNumber = i;
        strcpy(temp->patientName, "Available");
        temp->next = NULL;
        last->next = temp;
        last = temp;
    }
}

// Function to allocate a bed to a patient
void allocateBed(struct BedNode *p, int bedNumber, char patientName[])
{
    while (p != NULL)
    {
```

```
        if (p->bedNumber == bedNumber && strcmp(p->patientName, "Available") ==
0)
        {
            strcpy(p->patientName, patientName);   // Assign the bed to the patient
            printf("Bed %d allocated to %s\n", p->bedNumber, p->patientName);
            return;
        }
        p = p->next;
    }
    // If the bed is not found or not available
    printf("Bed %d is not available or invalid.\n", bedNumber);
}

// Function to display the current bed allocation
void displayBedAllocation(struct BedNode *p)
{
    if (p == NULL)
    {
        printf("No beds have been created.\n");
        return;
    }

    // Traverse through the list and display bed details
    printf("Current Bed Allocation:\n");
    while (p != NULL)
    {
        printf("Bed Number: %d, Patient: %s\n", p->bedNumber, p->patientName);
        p = p->next;
    }
}*/


// 3.
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct InventoryNode
{
    int itemID;
    char itemName[50];
    int quantity;
    struct InventoryNode *next;
} *first = NULL;

// Function prototypes
void createInventoryList(int itemCount);
void displayInventory(struct InventoryNode *p);
void insertInventoryItem(struct InventoryNode *p, int itemID, char itemName[], int
quantity);
```

```
int main()
{
    int itemCount = 3;
    createInventoryList(itemCount);

    printf("Initial Inventory List:\n");
    displayInventory(first);

    printf("\nAdding a new inventory item:\n");
    insertInventoryItem(first, 4, "Bandage", 200);
    displayInventory(first);

    return 0;
}

// Function to create an initial inventory list
void createInventoryList(int itemCount)
{
    int i;
    struct InventoryNode *temp, *last;

    // Create first inventory item
    first = (struct InventoryNode *)malloc(sizeof(struct InventoryNode));
    first->itemID = 1;
    strcpy(first->itemName, "Paracetamol");
    first->quantity = 50;
    first->next = NULL;
    last = first;

    // Create remaining inventory items
    for (i = 2; i <= itemCount; i++)
    {
        temp = (struct InventoryNode *)malloc(sizeof(struct InventoryNode));
        temp->itemID = i;


        if (i == 2)
            strcpy(temp->itemName, "Aspirin");
        else
            strcpy(temp->itemName, "Cough Syrup");

        temp->quantity = 100;
        temp->next = NULL;
        last->next = temp;
        last = temp;
    }
}
```

```
void insertInventoryItem(struct InventoryNode *p, int itemID, char itemName[], int
quantity)
{
    struct InventoryNode *temp, *last = p;


    while (last->next != NULL)
        last = last->next;


    temp = (struct InventoryNode *)malloc(sizeof(struct InventoryNode));
    temp->itemID = itemID;
    strcpy(temp->itemName, itemName);
    temp->quantity = quantity;
    temp->next = NULL;

    last->next = temp;
}

// Function to display the current inventory list
void displayInventory(struct InventoryNode *p)
{
    while (p != NULL)
    {
        printf("Item ID: %d, Item Name: %s, Quantity: %d\n", p->itemID, p->itemName,
p->quantity);
        p = p->next;
    }
}



//4.
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Define structure for appointments
struct AppointmentNode
{
    char patientName[50];     // Name of the patient
    char appointmentDate[20]; // Appointment date (e.g., "2025-01-15")
    char appointmentTime[20]; // Appointment time (e.g., "10:30 AM")
    struct AppointmentNode *next; // Pointer to the next appointment
} *first = NULL;

// Function prototypes
void createAppointmentList(int count);
void insertAppointment(struct AppointmentNode *p, char patientName[], char
appointmentDate[], char appointmentTime[]);
void displayAppointments(struct AppointmentNode *p);
```

```
int main()
{
    int count = 3;
    createAppointmentList(count);

    printf("Initial Appointment List:\n");
    displayAppointments(first);

    printf("\nAdding a new appointment:\n");
    insertAppointment(first, "John Smith", "2025-01-20", "11:00 AM");
    displayAppointments(first);

    return 0;
}

// Function to create an initial appointment list
void createAppointmentList(int count)
{
    int i;
    struct AppointmentNode *temp, *last;

    // Create the first appointment
    first = (struct AppointmentNode *)malloc(sizeof(struct AppointmentNode));
    strcpy(first->patientName, "Alice Brown");
    strcpy(first->appointmentDate, "2025-01-18");
    strcpy(first->appointmentTime, "9:30 AM");
    first->next = NULL;
    last = first;

    // Create remaining appointments
    for (i = 2; i <= count; i++)
    {
        temp = (struct AppointmentNode *)malloc(sizeof(struct AppointmentNode));
        if (i == 2)
        {
            strcpy(temp->patientName, "Bob White");
            strcpy(temp->appointmentDate, "2025-01-19");
            strcpy(temp->appointmentTime, "10:00 AM");
        }
        else
        {
            strcpy(temp->patientName, "Charlie Green");
            strcpy(temp->appointmentDate, "2025-01-19");
            strcpy(temp->appointmentTime, "10:30 AM");
        }
        temp->next = NULL;
        last->next = temp;
        last = temp;
    }
```

```
}

// Function to insert a new appointment
void insertAppointment(struct AppointmentNode *p, char patientName[], char
appointmentDate[], char appointmentTime[])
{
    struct AppointmentNode *temp, *last = p;

    // Traverse to the last node
    while (last->next != NULL)
        last = last->next;

    // Create a new node for the new appointment
    temp = (struct AppointmentNode *)malloc(sizeof(struct AppointmentNode));
    strcpy(temp->patientName, patientName);
    strcpy(temp->appointmentDate, appointmentDate);
    strcpy(temp->appointmentTime, appointmentTime);
    temp->next = NULL;

    // Link the new node to the last node
    last->next = temp;
}

// Function to display all scheduled appointments
void displayAppointments(struct AppointmentNode *p)
{
    if (p == NULL)
    {
        printf("No appointments scheduled.\n");
        return;
    }

    // Traverse through the list and display appointment details
    while (p != NULL)
    {
        printf("Patient: %s, Date: %s, Time: %s\n", p->patientName, p-
>appointmentDate, p->appointmentTime);
        p = p->next;
    }
}


//5.
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Define structure for emergency contact
struct EmergencyContact
{
```

```
    char name[50];
    char phoneNumber[15];
    struct EmergencyContact *next;
} *first = NULL;

// Function prototypes
void createContactList(char contacts[][2][50], int n);
void insertContact(struct EmergencyContact *p, char name[], char phoneNumber[]);
void displayContacts(struct EmergencyContact *p);

int main()
{
    char emergencyContacts[][2][50] = {{"John Doe", "123-456-7890"}, {"Jane Smith",
"987-654-3210"}};
    createContactList(emergencyContacts, 2);
    printf("Initial emergency contact list:\n");
    displayContacts(first);
    printf("\nAdding a new emergency contact:\n");
    insertContact(first, "Alex Brown", "555-555-5555");
    displayContacts(first);
    return 0;
}

void createContactList(char contacts[][2][50], int n)
{
    int i;
    struct EmergencyContact *temp, *last;
    first = (struct EmergencyContact *)malloc(sizeof(struct EmergencyContact));
    strcpy(first->name, contacts[0][0]);
    strcpy(first->phoneNumber, contacts[0][1]);
    first->next = NULL;
    last = first;

    for (i = 1; i < n; i++)
    {
        temp = (struct EmergencyContact *)malloc(sizeof(struct EmergencyContact));
        strcpy(temp->name, contacts[i][0]);
        strcpy(temp->phoneNumber, contacts[i][1]);
        temp->next = NULL;
        last->next = temp;
        last = temp;
    }
}

void insertContact(struct EmergencyContact *p, char name[], char phoneNumber[])
{
    struct EmergencyContact *temp, *last = p;
    temp = (struct EmergencyContact *)malloc(sizeof(struct EmergencyContact));
    strcpy(temp->name, name);
    strcpy(temp->phoneNumber, phoneNumber);
```

```
    temp->next = NULL;
    while (last->next != NULL)
        last = last->next;
    last->next = temp;
}

void displayContacts(struct EmergencyContact *p)
{
    while (p != NULL)
    {
        printf("Name: %s, Phone: %s\n", p->name, p->phoneNumber);
        p = p->next;
    }
}


//6.
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Define structure for surgery schedule
struct SurgeryNode
{
    char patientName[50];
    char surgeryType[50];
    char surgeryDate[20];
    struct SurgeryNode *next;
} *first = NULL;

// Function prototypes
void createSurgerySchedule(char schedules[][3][50], int n);
void insertSurgery(struct SurgeryNode *p, char patientName[], char surgeryType[],
char surgeryDate[]);
void displaySurgerySchedule(struct SurgeryNode *p);

int main()
{
    char surgerySchedules[][3][50] = {{"Alice Brown", "Appendectomy", "2025-02-15"},
{"Bob White", "Knee Replacement", "2025-02-16"}};
    createSurgerySchedule(surgerySchedules, 2);
    printf("Initial surgery schedule:\n");
    displaySurgerySchedule(first);
    printf("\nAdding a new surgery to the schedule:\n");
    insertSurgery(first, "Charlie Green", "Heart Bypass", "2025-02-17");
    displaySurgerySchedule(first);
    return 0;
}

void createSurgerySchedule(char schedules[][3][50], int n)
```

```
{
    int i;
    struct SurgeryNode *temp, *last;
    first = (struct SurgeryNode *)malloc(sizeof(struct SurgeryNode));
    strcpy(first->patientName, schedules[0][0]);
    strcpy(first->surgeryType, schedules[0][1]);
    strcpy(first->surgeryDate, schedules[0][2]);
    first->next = NULL;
    last = first;

    for (i = 1; i < n; i++)
    {
        temp = (struct SurgeryNode *)malloc(sizeof(struct SurgeryNode));
        strcpy(temp->patientName, schedules[i][0]);
        strcpy(temp->surgeryType, schedules[i][1]);
        strcpy(temp->surgeryDate, schedules[i][2]);
        temp->next = NULL;
        last->next = temp;
        last = temp;
    }
}

void insertSurgery(struct SurgeryNode *p, char patientName[], char surgeryType[],
char surgeryDate[])
{
    struct SurgeryNode *temp, *last = p;
    temp = (struct SurgeryNode *)malloc(sizeof(struct SurgeryNode));
    strcpy(temp->patientName, patientName);
    strcpy(temp->surgeryType, surgeryType);
    strcpy(temp->surgeryDate, surgeryDate);
    temp->next = NULL;
    while (last->next != NULL)
        last = last->next;
    last->next = temp;
}

void displaySurgerySchedule(struct SurgeryNode *p)
{
    while (p != NULL)
    {
        printf("Patient: %s, Surgery: %s, Date: %s\n", p->patientName, p->surgeryType,
p->surgeryDate);
        p = p->next;
    }
}

//7.
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
// Define structure for patient history record
struct PatientHistoryNode
{
    char patientName[50];
    char diagnosis[100];
    char treatment[100];
    struct PatientHistoryNode *next;
} *first = NULL;

// Function prototypes
void createHistoryRecordList(char records[][3][50], int n);
void insertHistoryRecord(struct PatientHistoryNode *p, char patientName[], char
diagnosis[], char treatment[]);
void displayHistoryRecords(struct PatientHistoryNode *p);

int main()
{
    char historyRecords[][3][50] = {{"Alice Brown", "Fever", "Paracetamol"}, {"Bob
White", "Knee Injury", "Surgery"}};
    createHistoryRecordList(historyRecords, 2);
    printf("Initial patient history records:\n");
    displayHistoryRecords(first);
    printf("\nAdding a new patient history record:\n");
    insertHistoryRecord(first, "Charlie Green", "Cold", "Cough Syrup");
    displayHistoryRecords(first);
    return 0;
}

void createHistoryRecordList(char records[][3][50], int n)
{
    int i;
    struct PatientHistoryNode *temp, *last;
    first = (struct PatientHistoryNode *)malloc(sizeof(struct PatientHistoryNode));
    strcpy(first->patientName, records[0][0]);
    strcpy(first->diagnosis, records[0][1]);
    strcpy(first->treatment, records[0][2]);
    first->next = NULL;
    last = first;

    for (i = 1; i < n; i++)
    {
        temp = (struct PatientHistoryNode *)malloc(sizeof(struct PatientHistoryNode));
        strcpy(temp->patientName, records[i][0]);
        strcpy(temp->diagnosis, records[i][1]);
        strcpy(temp->treatment, records[i][2]);
        temp->next = NULL;
        last->next = temp;
        last = temp;
    }
```

```
}

void insertHistoryRecord(struct PatientHistoryNode *p, char patientName[], char
diagnosis[], char treatment[])
{
    struct PatientHistoryNode *temp, *last = p;
    temp = (struct PatientHistoryNode *)malloc(sizeof(struct PatientHistoryNode));
    strcpy(temp->patientName, patientName);
    strcpy(temp->diagnosis, diagnosis);
    strcpy(temp->treatment, treatment);
    temp->next = NULL;
    while (last->next != NULL)
        last = last->next;
    last->next = temp;
}

void displayHistoryRecords(struct PatientHistoryNode *p)
{
    while (p != NULL)
    {
        printf("Patient: %s, Diagnosis: %s, Treatment: %s\n", p->patientName, p-
>diagnosis, p->treatment);
        p = p->next;
    }
}

//8.
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Define structure for medical test
struct MedicalTestNode
{
    char patientName[50];
    char testName[50];
    char testDate[20];
    struct MedicalTestNode *next;
} *first = NULL;

// Function prototypes
void createMedicalTestList(char tests[][3][50], int n);
void insertMedicalTest(struct MedicalTestNode *p, char patientName[], char
testName[], char testDate[]);
void displayMedicalTests(struct MedicalTestNode *p);

int main()
{
    char medicalTests[][3][50] = {{"Alice Brown", "Blood Test", "2025-02-01"}, {"Bob
White", "X-Ray", "2025-02-05"}};
```

```
    createMedicalTestList(medicalTests, 2);
    printf("Initial medical test list:\n");
    displayMedicalTests(first);
    printf("\nAdding a new medical test result:\n");
    insertMedicalTest(first, "Charlie Green", "MRI", "2025-02-10");
    displayMedicalTests(first);
    return 0;
}

void createMedicalTestList(char tests[][3][50], int n)
{
    int i;
    struct MedicalTestNode *temp, *last;
    first = (struct MedicalTestNode *)malloc(sizeof(struct MedicalTestNode));
    strcpy(first->patientName, tests[0][0]);
    strcpy(first->testName, tests[0][1]);
    strcpy(first->testDate, tests[0][2]);
    first->next = NULL;
    last = first;

    for (i = 1; i < n; i++)
    {
        temp = (struct MedicalTestNode *)malloc(sizeof(struct MedicalTestNode));
        strcpy(temp->patientName, tests[i][0]);
        strcpy(temp->testName, tests[i][1]);
        strcpy(temp->testDate, tests[i][2]);
        temp->next = NULL;
        last->next = temp;
        last = temp;
    }
}

void insertMedicalTest(struct MedicalTestNode *p, char patientName[], char
testName[], char testDate[])
{
    struct MedicalTestNode *temp, *last = p;
    temp = (struct MedicalTestNode *)malloc(sizeof(struct MedicalTestNode));
    strcpy(temp->patientName, patientName);
    strcpy(temp->testName, testName);
    strcpy(temp->testDate, testDate);
    temp->next = NULL;
    while (last->next != NULL)
        last = last->next;
    last->next = temp;
}

void displayMedicalTests(struct MedicalTestNode *p)
{
    while (p != NULL)
    {
```

```
        printf("Patient: %s, Test: %s, Date: %s\n", p->patientName, p->testName, p->testDate);
        p = p->next;
    }
}

//9.
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Define structure for prescription
struct PrescriptionNode
{
    char patientName[50];
    char medication[50];
    char dosage[50];
    struct PrescriptionNode *next;
} *first = NULL;

// Function prototypes
void createPrescriptionList(char prescriptions[][3][50], int n);
void insertPrescription(struct PrescriptionNode *p, char patientName[], char medication[], char dosage[]);
void displayPrescriptions(struct PrescriptionNode *p);

int main()
{
    char prescriptions[][3][50] = {{"Alice Brown", "Paracetamol", "500mg"}, {"Bob White", "Aspirin", "100mg"}};
    createPrescriptionList(prescriptions, 2);
    printf("Initial prescription list:\n");
    displayPrescriptions(first);
    printf("\nAdding a new prescription:\n");
    insertPrescription(first, "Charlie Green", "Cough Syrup", "10ml");
    displayPrescriptions(first);
    return 0;
}

void createPrescriptionList(char prescriptions[][3][50], int n)
{
    int i;
    struct PrescriptionNode *temp, *last;
    first = (struct PrescriptionNode *)malloc(sizeof(struct PrescriptionNode));
    strcpy(first->patientName, prescriptions[0][0]);
    strcpy(first->medication, prescriptions[0][1]);
    strcpy(first->dosage, prescriptions[0][2]);
    first->next = NULL;
    last = first;
```

```
    for (i = 1; i < n; i++)
    {
        temp = (struct PrescriptionNode *)malloc(sizeof(struct PrescriptionNode));
        strcpy(temp->patientName, prescriptions[i][0]);
        strcpy(temp->medication, prescriptions[i][1]);
        strcpy(temp->dosage, prescriptions[i][2]);
        temp->next = NULL;
        last->next = temp;
        last = temp;
    }
}

void insertPrescription(struct PrescriptionNode *p, char patientName[], char
medication[], char dosage[])
{
    struct PrescriptionNode *temp, *last = p;
    temp = (struct PrescriptionNode *)malloc(sizeof(struct PrescriptionNode));
    strcpy(temp->patientName, patientName);
    strcpy(temp->medication, medication);
    strcpy(temp->dosage, dosage);
    temp->next = NULL;
    while (last->next != NULL)
        last = last->next;
    last->next = temp;
}

void displayPrescriptions(struct PrescriptionNode *p)
{
    while (p != NULL)
    {
        printf("Patient: %s, Medication: %s, Dosage: %s\n", p->patientName, p-
>medication, p->dosage);
        p = p->next;
    }
}

// 10.
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Define structure for hospital staff
struct StaffNode
{
    char name[50];
    char position[50];
    struct StaffNode *next;
} *first = NULL;

// Function prototypes
```

```
void createStaffRoster(char staff[][2][50], int n);
void insertStaffMember(struct StaffNode *p, char name[], char position[]);
void displayStaffRoster(struct StaffNode *p);

int main()
{
    char staffRoster[][2][50] = {{"Dr. Smith", "Surgeon"}, {"Nurse Mary", "Nurse"}};
    createStaffRoster(staffRoster, 2);
    printf("Initial hospital staff roster:\n");
    displayStaffRoster(first);
    printf("\nAdding a new staff member:\n");
    insertStaffMember(first, "Dr. John", "Cardiologist");
    displayStaffRoster(first);
    return 0;
}

void createStaffRoster(char staff[][2][50], int n)
{
    int i;
    struct StaffNode *temp, *last;
    first = (struct StaffNode *)malloc(sizeof(struct StaffNode));
    strcpy(first->name, staff[0][0]);
    strcpy(first->position, staff[0][1]);
    first->next = NULL;
    last = first;

    for (i = 1; i < n; i++)
    {
        temp = (struct StaffNode *)malloc(sizeof(struct StaffNode));
        strcpy(temp->name, staff[i][0]);
        strcpy(temp->position, staff[i][1]);
        temp->next = NULL;
        last->next = temp;
        last = temp;
    }
}

void insertStaffMember(struct StaffNode *p, char name[], char position[])
{
    struct StaffNode *temp, *last = p;
    temp = (struct StaffNode *)malloc(sizeof(struct StaffNode));
    strcpy(temp->name, name);
    strcpy(temp->position, position);
    temp->next = NULL;
    while (last->next != NULL)
        last = last->next;
    last->next = temp;
}

void displayStaffRoster(struct StaffNode *p)
```

```
{
    while (p != NULL)
    {
        printf("Name: %s, Position: %s\n", p->name, p->position);
        p = p->next;
    }
}
```