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

#define MAX_PATIENTS 100

// Structure to hold patient data
typedef struct {
    int id;
    char name[100];
    int age;
    char gender[10];
} Patient;

// Node structure for appointments
typedef struct Appointment {
    int patientId;
    char date[20];
    char time[10];
    struct Appointment* next;
} Appointment;

Patient patients[MAX_PATIENTS];
int patientCount = 0;

Appointment* head = NULL;

// Function to add a new patient
void addPatient() {
    if (patientCount >= MAX_PATIENTS) {
        printf("Patient list is full.\n");
```

```
        return;
    }

    Patient p;
    p.id = patientCount + 1;

    printf("Enter patient name: ");
    scanf(" %[^\n]", p.name);

    printf("Enter patient age: ");
    scanf("%d", &p.age);

    printf("Enter patient gender: ");
    scanf(" %s", p.gender);

    patients[patientCount++] = p;

    printf("Patient added successfully with ID %d.\n", p.id);
}

// Function to display all patients
void displayPatients() {
    printf("\n--- Patient List ---\n");
    for (int i = 0; i < patientCount; i++) {
        printf("ID: %d, Name: %s, Age: %d, Gender: %s\n",
               patients[i].id, patients[i].name, patients[i].age, patients[i].gender);
    }
}

// Function to add an appointment
void addAppointment() {
```

```
if (patientCount == 0) {
    printf("No patients available. Add patients first.\n");
    return;
}

int id;
char date[20];
char time[10];

printf("Enter patient ID for appointment: ");
scanf("%d", &id);

int found = 0;
for (int i = 0; i < patientCount; i++) {
    if (patients[i].id == id) {
        found = 1;
        break;
    }
}

if (!found) {
    printf("Patient ID not found.\n");
    return;
}

Appointment* newApp = (Appointment*)malloc(sizeof(Appointment));
newApp->patientId = id;

printf("Enter appointment date (YYYY-MM-DD): ");
scanf("%s", newApp->date);
```

```
    printf("Enter appointment time (HH:MM): ");
    scanf("%s", newApp->time);

    newApp->next = NULL;

    if (head == NULL) {
        head = newApp;
    } else {
        Appointment* temp = head;
        while (temp->next != NULL)
            temp = temp->next;
        temp->next = newApp;
    }

    printf("Appointment added successfully.\n");
}

// Function to display all appointments
void displayAppointments() {
    printf("\n--- Appointments List ---\n");
    Appointment* temp = head;

    if (temp == NULL) {
        printf("No appointments scheduled.\n");
        return;
    }

    while (temp != NULL) {
        printf("Patient ID: %d, Date: %s, Time: %s\n", temp->patientId, temp->date, temp->time);
        temp = temp->next;
    }
```

```
}

// Function to clean up memory
void freeAppointments() {
    Appointment* temp;
    while (head != NULL) {
        temp = head;
        head = head->next;
        free(temp);
    }
}

// Main menu
int main() {
    int choice;

    do {
        printf("\n--- Hospital Management System ---\n");
        printf("1. Add Patient\n");
        printf("2. Display Patients\n");
        printf("3. Add Appointment\n");
        printf("4. Display Appointments\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

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

```
            displayPatients();
            break;
        case 3:
            addAppointment();
            break;
        case 4:
            displayAppointments();
            break;
        case 5:
            freeAppointments();
            printf("Exiting system.\n");
            break;
        default:
            printf("Invalid choice. Try again.\n");
      }

   } while (choice != 5);

   return 0;
}
```