Problem Statement 1: Bus Seat Allocation System

Design and implement a C program to manage bus seat allocations for Redbus. The
program should have the following features:

- Accept bus details (bus number, route, departure time, and number of seats)
- Accept passenger details (name, age, and contact number)
- Allocate seats to passengers based on availability
- Display allocated seat numbers and passenger details
- Handle cancellations and re-allocate seats accordingly

Implement functions for seat allocation, cancellation, and display of allocated seats.

Code:

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

#define MAX_SEATS 50
#define MAX_PASSENGERS 100

typedef struct {
   char name[50];
   int age;
   char contact[15];
   int seatNumber;
} Passenger;

typedef struct {
   char busNumber[20];
   char route[100];
   char departureTime[20];
   int totalSeats;
```

```
    int availableSeats;
    Passenger passengers[MAX_PASSENGERS];
} Bus;

Bus bus;

void initializeBus() {
    printf("Enter Bus Number: ");
    scanf("%s", bus.busNumber);
    printf("Enter Route: ");
    scanf(" %[^\n]", bus.route);
    printf("Enter Departure Time: ");
    scanf("%s", bus.departureTime);
    printf("Enter Total Number of Seats: ");
    scanf("%d", &bus.totalSeats);
    bus.availableSeats = bus.totalSeats;
}

void allocateSeat() {
    if (bus.availableSeats <= 0) {
        printf("No seats available!\n");
        return;
    }

    Passenger newPassenger;
    printf("Enter Passenger Name: ");
    scanf(" %[^\n]", newPassenger.name);
    printf("Enter Passenger Age: ");
    scanf("%d", &newPassenger.age);
    printf("Enter Passenger Contact Number: ");
    scanf("%s", newPassenger.contact);
```

```
    // Find the first available seat
    for (int i = 0; i < MAX_PASSENGERS; i++) {
        if (bus.passengers[i].seatNumber == 0) { // Seat is available
            newPassenger.seatNumber = i + 1; // Seat numbers start from 1
            bus.passengers[i] = newPassenger;
            bus.availableSeats--;
            printf("Seat allocated: %d\n", newPassenger.seatNumber);
            return;
        }
    }
}

void cancelSeat() {
    int seatNumber;
    printf("Enter Seat Number to Cancel: ");
    scanf("%d", &seatNumber);

    if (seatNumber < 1 || seatNumber > MAX_PASSENGERS ||
bus.passengers[seatNumber - 1].seatNumber == 0) {
        printf("Invalid seat number or seat not allocated!\n");
        return;
    }

    // Free the seat
    bus.passengers[seatNumber - 1].seatNumber = 0;
    bus.availableSeats++;
    printf("Seat number %d cancelled successfully.\n", seatNumber);
}

void displayAllocatedSeats() {
    printf("Allocated Seats:\n");
    for (int i = 0; i < MAX_PASSENGERS; i++) {
```

```
        if (bus.passengers[i].seatNumber != 0) {
            printf("Seat Number: %d, Name: %s, Age: %d, Contact: %s\n",
                   bus.passengers[i].seatNumber,
                   bus.passengers[i].name,
                   bus.passengers[i].age,
                   bus.passengers[i].contact);
        }
    }
}

int main() {
    int choice;

    initializeBus();

    do {
        printf("\nBus Seat Allocation System\n");
        printf("1. Allocate Seat\n");
        printf("2. Cancel Seat\n");
        printf("3. Display Allocated Seats\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                allocateSeat();
                break;
            case 2:
                cancelSeat();
                break;
            case 3:
```

```
            displayAllocatedSeats();
            break;
        case 4:
            printf("Exiting...\n");
            break;
        default:
            printf("Invalid choice! Please try again.\n");
    }
  } while (choice != 4);

  return 0;
}
```

Problem Statement 2: Route Optimization System
Develop a C program to optimize bus routes for Redbus. The program should:
1. Accept route details (source, destination, and intermediate stops)
2. Calculate the shortest route between two points using a suitable algorithm (e.g.,
Dijkstra's or Floyd-Warshall)
3. Display the optimized route and estimated travel time
4. Handle changes in route details and re-optimize the route accordingly
Implement functions for route calculation, optimization, and display

CODE:

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

#define MAX_NODES 100
#define INF INT_MAX

typedef struct {
   char name[50];
} Node;

typedef struct {
   int graph[MAX_NODES][MAX_NODES];
   Node nodes[MAX_NODES];
   int nodeCount;
} Graph;

void initGraph(Graph *g) {
   g->nodeCount = 0;
   for (int i = 0; i < MAX_NODES; i++) {
```

```
        for (int j = 0; j < MAX_NODES; j++) {
            g->graph[i][j] = (i == j) ? 0 : INF;
        }
    }
}

int addNode(Graph *g, const char *name) {
    if (g->nodeCount >= MAX_NODES) {
        return -1; // Graph is full
    }
    strcpy(g->nodes[g->nodeCount].name, name);
    return g->nodeCount++;
}

void addEdge(Graph *g, int src, int dest, int weight) {
    g->graph[src][dest] = weight;
    g->graph[dest][src] = weight; // Assuming undirected graph
}

void dijkstra(Graph *g, int start, int end) {
    int dist[MAX_NODES];
    int prev[MAX_NODES];
    int visited[MAX_NODES] = {0};

    for (int i = 0; i < g->nodeCount; i++) {
        dist[i] = INF;
        prev[i] = -1;
    }
    dist[start] = 0;

    for (int i = 0; i < g->nodeCount; i++) {
        int minDist = INF;
```

```
        int u = -1;

        for (int j = 0; j < g->nodeCount; j++) {
            if (!visited[j] && dist[j] < minDist) {
                minDist = dist[j];
                u = j;
            }
        }

        if (u == -1) break; // All reachable nodes are visited
        visited[u] = 1;

        for (int v = 0; v < g->nodeCount; v++) {
            if (g->graph[u][v] != INF && !visited[v]) {
                int alt = dist[u] + g->graph[u][v];
                if (alt < dist[v]) {
                    dist[v] = alt;
                    prev[v] = u;
                }
            }
        }
    }

    // Display the shortest path
    printf("Shortest path from %s to %s:\n", g->nodes[start].name, g-
>nodes[end].name);
    int path[MAX_NODES];
    int pathIndex = 0;
    for (int at = end; at != -1; at = prev[at]) {
        path[pathIndex++] = at;
    }
    for (int i = pathIndex - 1; i >= 0; i--) {
```

```
        printf("%s", g->nodes[path[i]].name);
        if (i > 0) printf(" -> ");
    }
    printf("\nEstimated travel time: %d\n", dist[end]);
}

int main() {
    Graph g;
    initGraph(&g);

    // Example nodes
    int src = addNode(&g, "A");
    int dest = addNode(&g, "B");
    int inter1 = addNode(&g, "C");
    int inter2 = addNode(&g, "D");

    // Example edges (weights represent travel time)
    addEdge(&g, src, inter1, 5);
    addEdge(&g, inter1, inter2, 10);
    addEdge(&g, inter2, dest, 2);
    addEdge(&g, src, dest, 20);

    // Calculate and display the shortest route
    dijkstra(&g, src, dest);

    return 0;
}
```

Problem Statement 3: Ticket Booking and Payment System
Design a C program to manage ticket bookings and payments for Redbus. The program
should:
1. Accept ticket booking details (passenger name, age, contact number, and travel dates)
2. Calculate ticket prices based on travel dates and bus type
3. Accept payment details (payment method, amount, and transaction ID)
4. Display ticket booking confirmation and payment receipt
5. Handle cancellations and refunds accordingly
Implement functions for ticket booking, payment processing, and cancellation.

CODE:

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

#define MAX_BOOKINGS 100

typedef struct {
    char passengerName[50];
    int age;
    char contactNumber[15];
    char travelDate[11]; // Format: YYYY-MM-DD
    char busType[20]; // e.g., "Sleeper", "Seater"
    float ticketPrice;
    int isCancelled;
} TicketBooking;

typedef struct {
    char paymentMethod[20]; // e.g., "Credit Card", "Debit Card", "Net Banking"
    float amount;
```

```
    char transactionID[20];
} Payment;

TicketBooking bookings[MAX_BOOKINGS];
Payment payments[MAX_BOOKINGS];
int bookingCount = 0;

float calculateTicketPrice(char *travelDate, char *busType) {
    // Simple pricing logic based on bus type and travel date
    float basePrice = 100.0; // Base price
    if (strcmp(busType, "Sleeper") == 0) {
        basePrice += 50.0; // Extra for sleeper
    }
    // Add more pricing logic based on travel date if needed
    return basePrice;
}

void bookTicket() {
    if (bookingCount >= MAX_BOOKINGS) {
        printf("Booking limit reached. Cannot book more tickets.\n");
        return;
    }

    TicketBooking booking;
    printf("Enter passenger name: ");
    scanf("%s", booking.passengerName);
    printf("Enter age: ");
    scanf("%d", &booking.age);
    printf("Enter contact number: ");
    scanf("%s", booking.contactNumber);
    printf("Enter travel date (YYYY-MM-DD): ");
    scanf("%s", booking.travelDate);
```

```
    printf("Enter bus type (Sleeper/Seater): ");
    scanf("%s", booking.busType);

    booking.ticketPrice = calculateTicketPrice(booking.travelDate,
booking.busType);
    booking.isCancelled = 0; // Not cancelled by default

    bookings[bookingCount] = booking;
    bookingCount++;

    printf("Ticket booked successfully! Ticket Price: %.2f\n", booking.ticketPrice);
}

void processPayment(int bookingIndex) {
    if (bookingIndex < 0 || bookingIndex >= bookingCount ||
bookings[bookingIndex].isCancelled) {
        printf("Invalid booking index or booking is cancelled.\n");
        return;
    }

    Payment payment;
    printf("Enter payment method: ");
    scanf("%s", payment.paymentMethod);
    printf("Enter amount: ");
    scanf("%f", &payment.amount);
    printf("Enter transaction ID: ");
    scanf("%s", payment.transactionID);

    // Check if payment amount matches ticket price
    if (payment.amount != bookings[bookingIndex].ticketPrice) {
        printf("Payment amount does not match ticket price. Payment failed.\n");
        return;
```

```
    }

    payments[bookingIndex] = payment;
    printf("Payment processed successfully! Transaction ID: %s\n",
payment.transactionID);
}

void cancelBooking(int bookingIndex) {
    if (bookingIndex < 0 || bookingIndex >= bookingCount ||
bookings[bookingIndex].isCancelled) {
        printf("Invalid booking index or booking is already cancelled.\n");
        return;
    }

    bookings[bookingIndex].isCancelled = 1; // Mark as cancelled
    printf("Booking cancelled successfully.\n");
}

void displayBookingConfirmation(int bookingIndex) {
    if (bookingIndex < 0 || bookingIndex >= bookingCount) {
        printf("Invalid booking index.\n");
        return;
    }

    TicketBooking booking = bookings[bookingIndex];
    if (booking.isCancelled) {
        printf("This booking has been cancelled.\n");
        return;
    }

    printf("\n--- Booking Confirmation ---\n");
    printf("Passenger Name: %s\n", booking.passengerName);
```

```
    printf("Age: %d\n", booking.age);
    printf("Contact Number: %s\n", booking.contactNumber);
    printf("Travel Date: %s\n", booking.travelDate);
    printf("Bus Type: %s\n", booking.busType);
    printf("Ticket Price: %.2f\n", booking.ticketPrice);
    printf("--------------------------\n");

    // Display payment receipt if payment is processed
    if (strlen(payments[bookingIndex].transactionID) > 0) {
        printf("--- Payment Receipt ---\n");
        printf("Payment Method: %s\n", payments[bookingIndex].paymentMethod);
        printf("Amount: %.2f\n", payments[bookingIndex].amount);
        printf("Transaction ID: %s\n", payments[bookingIndex].transactionID);
        printf("----------------------\n");
    }
}

int main() {
    int choice, bookingIndex;

    while (1) {
        printf("\n--- Redbus Ticket Booking System ---\n");
        printf("1. Book Ticket\n");
        printf("2. Process Payment\n");
        printf("3. Cancel Booking\n");
        printf("4. Display Booking Confirmation\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
```

```
            bookTicket();
            break;
        case 2:
            printf("Enter booking index to process payment: ");
            scanf("%d", &bookingIndex);
            processPayment(bookingIndex);
            break;
        case 3:
            printf("Enter booking index to cancel: ");
            scanf("%d", &bookingIndex);
            cancelBooking(bookingIndex);
            break;
        case 4:
            printf("Enter booking index to display confirmation: ");
            scanf("%d", &bookingIndex);
            displayBookingConfirmation(bookingIndex);
            break;
        case 5:
            printf("Exiting the system. Thank you!\n");
            exit(0);
        default:
            printf("Invalid choice. Please try again.\n");
      }
   }

   return 0;
}
```

Problem Statement 4: Bus Tracking and Location System
Develop a C program to track bus locations and provide real-time updates for Redbus. The
program should:
1. Accept bus location updates (GPS coordinates and timestamp)
2. Calculate estimated arrival times based on bus location and route details
3. Display real-time bus location and estimated arrival times
4. Handle changes in bus location and route details accordingly
CODE:

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

#define MAX_BUSES 100

typedef struct {
   char bus_id[20];
   double latitude;
   double longitude;
   time_t timestamp; // Time of the location update
   double speed; // Speed in km/h
   double distance_to_destination; // Distance in km to the destination
} Bus;

Bus buses[MAX_BUSES];
int bus_count = 0;

// Function to update bus location
void update_bus_location(const char* bus_id, double latitude, double longitude,
double speed, double distance_to_destination) {
   for (int i = 0; i < bus_count; i++) {
```

```
        if (strcmp(buses[i].bus_id, bus_id) == 0) {
            buses[i].latitude = latitude;
            buses[i].longitude = longitude;
            buses[i].speed = speed;
            buses[i].distance_to_destination = distance_to_destination;
            buses[i].timestamp = time(NULL);
            return;
        }
    }

    // If bus not found, add a new bus
    if (bus_count < MAX_BUSES) {
        strcpy(buses[bus_count].bus_id, bus_id);
        buses[bus_count].latitude = latitude;
        buses[bus_count].longitude = longitude;
        buses[bus_count].speed = speed;
        buses[bus_count].distance_to_destination = distance_to_destination;
        buses[bus_count].timestamp = time(NULL);
        bus_count++;
    } else {
        printf("Bus limit reached. Cannot add more buses.\n");
    }
}

// Function to calculate estimated arrival time
double calculate_estimated_arrival_time(double speed, double distance) {
    if (speed <= 0) return -1; // Avoid division by zero
    return distance / speed * 60; // Return time in minutes
}

// Function to display bus information
void display_buses() {
```

```
    printf("Current Bus Locations and Estimated Arrival Times:\n");
    for (int i = 0; i < bus_count; i++) {
        double eta = calculate_estimated_arrival_time(buses[i].speed,
buses[i].distance_to_destination);
        printf("Bus ID: %s\n", buses[i].bus_id);
        printf("Location: (%.6f, %.6f)\n", buses[i].latitude, buses[i].longitude);
        printf("Last Update: %s", ctime(&buses[i].timestamp));
        if (eta >= 0) {
            printf("Estimated Arrival Time: %.2f minutes\n", eta);
        } else {
            printf("Estimated Arrival Time: Not available (speed is zero)\n");
        }
        printf("\n");
    }
}

int main() {
    int choice;
    char bus_id[20];
    double latitude, longitude, speed, distance_to_destination;

    while (1) {
        printf("1. Update Bus Location\n");
        printf("2. Display Bus Information\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter Bus ID: ");
                scanf("%s", bus_id);
```

```
            printf("Enter Latitude: ");
            scanf("%lf", &latitude);
            printf("Enter Longitude: ");
            scanf("%lf", &longitude);
            printf("Enter Speed (km/h): ");
            scanf("%lf", &speed);
            printf("Enter Distance to Destination (km): ");
            scanf("%lf", &distance_to_destination);
            update_bus_location(bus_id, latitude, longitude, speed,
distance_to_destination);
            break;
        case 2:
            display_buses();
            break;
        case 3:
            exit(0);
        default:
            printf("Invalid choice. Please try again.\n");
      }
   }

   return 0;
}
```