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

// Constants for the system
#define MAX_FLIGHTS 10
#define MAX_BUSINESS_SEATS 50
#define MAX_ECONOMY_SEATS 100
#define MAX_NAME_LENGTH 50
#define MAX_DESTINATION_LENGTH 50
#define MAX_RESERVATIONS 500

// Structure to represent a Flight
typedef struct {
    int flightNumber;
    char destination[MAX_DESTINATION_LENGTH];
    float businessPrice;
    float economyPrice;
    int availableBusinessSeats;
    int availableEconomySeats;
    int businessSeats[MAX_BUSINESS_SEATS]; // 0 for available, 1 for reserved
    int economySeats[MAX_ECONOMY_SEATS];   // 0 for available, 1 for reserved
} Flight;

// Structure to represent a Reservation
typedef struct {
    char passengerName[MAX_NAME_LENGTH];
    int flightNumber;
    char destination[MAX_DESTINATION_LENGTH];
    int seatNumber;
    int classType; // 1 for Business, 2 for Economy
    float price;
} Reservation;

// Global variables
Flight flights[MAX_FLIGHTS];
Reservation reservations[MAX_RESERVATIONS];
int *flightCount = NULL;
int *reservationCount = NULL;

// Function prototypes
void initializeSystem();
void displayMainMenu();
void displayAdminMenu();
void displayCustomerMenu(char *customerName);
void addFlight();
void displaySeatAvailability();
void displayAllReservations();
void displayTicketPrices();
void searchFlightsByDestination(char *customerName);
void reserveSeat(char *customerName);
void cancelReservation(char *customerName);
void checkTotalAmountSpent(char *customerName);
int findFlightByNumber(int flightNumber);
int findFirstAvailableSeat(int flightIndex, int classType);

// Main function
int main() {
    // Initialize the system
    initializeSystem();

    int mainChoice;
    char customerName[MAX_NAME_LENGTH];

    do {
        // Display main menu and get user choice
        printf("Select role (1 for Admin, 2 for Customer, 0 to Exit): ");
        scanf("%d", &mainChoice);

        switch (mainChoice) {
            case 1: // Admin
                displayAdminMenu();
                break;
            case 2: // Customer
                printf("Enter your name: ");
                scanf(" %[^\n]", customerName); // Read name with spaces
                displayCustomerMenu(customerName);
                break;
            case 0: // Exit
                printf("Exiting the system. Goodbye!\n");
                break;
            default:
                printf("Invalid role. Please select again.\n");
        }
    } while (mainChoice != 0);
    // Free allocated memory before exiting
    free(flightCount);
    free(reservationCount);

    return 0;
}

// Initialize the system
void initializeSystem() {
    // Allocate memory for the count variables
    flightCount = (int *)malloc(sizeof(int));
    reservationCount = (int *)malloc(sizeof(int));

    // Initialize counts to zero
    *flightCount = 0;
    *reservationCount = 0;

    // Initialize all flight seats as available (0)
    for (int i = 0; i < MAX_FLIGHTS; i++) {
        for (int j = 0; j < MAX_BUSINESS_SEATS; j++) {
            flights[i].businessSeats[j] = 0;
        }
        for (int j = 0; j < MAX_ECONOMY_SEATS; j++) {
            flights[i].economySeats[j] = 0;
        }
    }
}

// Display the admin menu and handle admin operations
void displayAdminMenu() {
    int adminChoice;

    do {
        printf("\nAdmin Menu:\n");
        printf("1. Add a flight\n");
        printf("2. Display seat availability\n");
        printf("3. Display all reservations\n");
        printf("4. Display ticket prices for each flight\n");
        printf("5. Exit to main menu\n");
        printf("Enter your choice: ");
        scanf("%d", &adminChoice);

        switch (adminChoice) {
            case 1:
                addFlight();
                break;
            case 2:
                displaySeatAvailability();
                break;
            case 3:
                displayAllReservations();
                break;
            case 4:
                displayTicketPrices();
                break;
            case 5:
                printf("Returning to main menu...\n");
                break;
            default:
                printf("Invalid choice! Please try again.\n");
        }
    } while (adminChoice != 5);
}

// Display the customer menu and handle customer operations
void displayCustomerMenu(char *customerName) {
    int customerChoice;

    do {
        printf("\nCustomer Menu:\n");
        printf("1. Search for flights by destination\n");
        printf("2. Reserve a seat\n");
        printf("3. Cancel a reservation\n");
        printf("4. Check total amount spent\n");
        printf("5. Exit to main menu\n");
        printf("Enter your choice: ");
        scanf("%d", &customerChoice);

        switch (customerChoice) {
            case 1:
                searchFlightsByDestination(customerName);
                break;
            case 2:
                reserveSeat(customerName);
                break;
            case 3:
                cancelReservation(customerName);
                break;
            case 4:
                checkTotalAmountSpent(customerName);
                break;
            case 5:
                printf("Returning to main menu...\n");
                break;
            default:
                printf("Invalid choice! Please try again.\n");
        }
    } while (customerChoice != 5);
}

// Add a new flight (Admin function) - Optimized with binary search
void addFlight() {
    if (*flightCount >= MAX_FLIGHTS) {
        printf("No more flights can be added.\n");
        return;
    }

    Flight newFlight;

    // Input flight details
    printf("Enter Flight Number: ");
    scanf("%d", &newFlight.flightNumber);

    // Binary search to check if flight number already exists
    int left = 0;
    int right = *flightCount - 1;
    int found = 0;

    // Sort flights by flight number first (insertion sort)
    for (int i = 1; i < *flightCount; i++) {
        Flight key = flights[i];
        int j = i - 1;

        while (j >= 0 && flights[j].flightNumber > key.flightNumber) {
            flights[j + 1] = flights[j];
            j--;
        }
        flights[j + 1] = key;
    }
    // Binary search for flight number
    while (left <= right) {
        int mid = left + (right - left) / 2;

        if (flights[mid].flightNumber == newFlight.flightNumber) {
            found = 1;
            break;
        }

        if (flights[mid].flightNumber < newFlight.flightNumber)
            left = mid + 1;
        else
            right = mid - 1;
    }

    if (found) {
        printf("Flight number already exists. Please use a different 
number.\n");
        return;
    }

    printf("Enter Destination: ");
    scanf(" %[^\n]", newFlight.destination);

    printf("Enter Business Class Ticket Price: ");
    scanf("%f", &newFlight.businessPrice);

    printf("Enter Economy Class Ticket Price: ");
    scanf("%f", &newFlight.economyPrice);

    // Initialize available seats
    newFlight.availableBusinessSeats = MAX_BUSINESS_SEATS;
    newFlight.availableEconomySeats = MAX_ECONOMY_SEATS;

    // Initialize all seats as available (0)
    for (int i = 0; i < MAX_BUSINESS_SEATS; i++) {
        newFlight.businessSeats[i] = 0;
    }
    for (int i = 0; i < MAX_ECONOMY_SEATS; i++) {
        newFlight.economySeats[i] = 0;
    }
    // Add the new flight to the array
    flights[*flightCount] = newFlight;
    (*flightCount)++;

    // Sort flights by flight number (insertion sort)
    for (int i = 1; i < *flightCount; i++) {
        Flight key = flights[i];
        int j = i - 1;

        while (j >= 0 && flights[j].flightNumber > key.flightNumber) {
            flights[j + 1] = flights[j];
            j--;
        }
        flights[j + 1] = key;
    }

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

// Display seat availability for all flights (Admin function)
void displaySeatAvailability() {
    if (*flightCount == 0) {
        printf("No flights to display.\n");
        return;
    }

    // Sort flights by available seats (bubble sort)
    for (int i = 0; i < *flightCount - 1; i++) {
        for (int j = 0; j < *flightCount - i - 1; j++) {
            if (flights[j].availableBusinessSeats +
flights[j].availableEconomySeats <
                flights[j + 1].availableBusinessSeats + flights[j +
1].availableEconomySeats) {
                Flight temp = flights[j];
                flights[j] = flights[j + 1];
                flights[j + 1] = temp;
            }
        }
    }

    printf("\nFlight Number  Destination      Business Seats  Economy
Seats\n");
    for (int i = 0; i < *flightCount; i++) {
        printf("%-14d %-16s %-15d %-13d\n",
               flights[i].flightNumber,
               flights[i].destination,
               flights[i].availableBusinessSeats,
               flights[i].availableEconomySeats);
    }
}

// Display all reservations (Admin function)
void displayAllReservations() {
    if (*reservationCount == 0) {
        printf("No reservations to display.\n");
        return;
    }

    // Sort reservations by flight number (selection sort)
    for (int i = 0; i < *reservationCount - 1; i++) {
        int min_idx = i;
        for (int j = i + 1; j < *reservationCount; j++) {
            if (reservations[j].flightNumber <
reservations[min_idx].flightNumber) {
                min_idx = j;
            }
        }
        if (min_idx != i) {
            Reservation temp = reservations[i];
            reservations[i] = reservations[min_idx];
            reservations[min_idx] = temp;
        }
    }

    printf("\nPassenger Name       Flight Number  Destination      Seat Number
1=Business, 2=Economy\n");
    for (int i = 0; i < *reservationCount; i++) {
        printf("%-20s %-14d %-16s %-12d %-5d\n",
               reservations[i].passengerName,
               reservations[i].flightNumber,
               reservations[i].destination,
               reservations[i].seatNumber,
               reservations[i].classType);
    }
}
```

```
// Display ticket prices for each flight (Admin function)
void displayTicketPrices() {
    if (*flightCount == 0) {
        printf("No flights to display ticket prices.\n");
        return;
    }

    // Sort flights by business price (insertion sort)
    for (int i = 1; i < *flightCount; i++) {
        Flight key = flights[i];
        int j = i - 1;

        while (j >= 0 && flights[j].businessPrice > key.businessPrice) {
            flights[j + 1] = flights[j];
            j--;
        }
        flights[j + 1] = key;
    }

    printf("\nFlight Number  Destination      Business Price  Economy
Price\n");
    for (int i = 0; i < *flightCount; i++) {
        printf("%-14d %-16s %-15.2f %-13.2f\n",
               flights[i].flightNumber,
               flights[i].destination,
               flights[i].businessPrice,
               flights[i].economyPrice);
    }
}

// Search for flights by destination (Customer function) - Optimized with
binary search
void searchFlightsByDestination(char *customerName) {
    if (*flightCount == 0) {
        printf("No flights available.\n");
        return;
    }

    char destination[MAX_DESTINATION_LENGTH];
    printf("Enter destination: ");
    scanf(" %[^\n]", destination);

    // Sort flights by destination for binary search
```

```c
    for (int i = 1; i < *flightCount; i++) {
        Flight key = flights[i];
        int j = i - 1;

        while (j >= 0 && strcasecmp(flights[j].destination, key.destination) >
0) {
            flights[j + 1] = flights[j];
            j--;
        }
        flights[j + 1] = key;
    }

    // Binary search (with adjustments for string comparison)
    int left = 0;
    int right = *flightCount - 1;
    int found = 0;

    while (left <= right) {
        int mid = left + (right - left) / 2;
        int cmp = strcasecmp(flights[mid].destination, destination);

        if (cmp == 0) {
            // Found a match at mid, but need to check for other matches due to
case-insensitivity
            found = 1;

            // Display the found flight
            printf("Flight Number: %d, Available Business Seats: %d, Available
Economy Seats: %d\n",
                flights[mid].flightNumber,
                flights[mid].availableBusinessSeats,
                flights[mid].availableEconomySeats);

            // Check for other matches to the left
            int idx = mid - 1;
            while (idx >= 0 && strcasecmp(flights[idx].destination,
destination) == 0) {
                printf("Flight Number: %d, Available Business Seats: %d,
Available Economy Seats: %d\n",
                    flights[idx].flightNumber,
                    flights[idx].availableBusinessSeats,
                    flights[idx].availableEconomySeats);
                idx--;
            }

            // Check for other matches to the right
            idx = mid + 1;
            while (idx < *flightCount && strcasecmp(flights[idx].destination, 
destination) == 0) {
                printf("Flight Number: %d, Available Business Seats: %d, 
Available Economy Seats: %d\n",
                    flights[idx].flightNumber,
                    flights[idx].availableBusinessSeats,
                    flights[idx].availableEconomySeats);
                idx++;
            }

            break;
        }

        if (cmp < 0) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }

    if (!found) {
        printf("No flights found for the destination %s.\n", destination);
    }
}

// Reserve a seat on a flight (Customer function)
void reserveSeat(char *customerName) {
    if (*flightCount == 0) {
        printf("No flights available.\n");
        return;
    }

    // Get destination from customer
    char destination[MAX_DESTINATION_LENGTH];
    printf("Enter destination: ");
    scanf(" %[^\n]", destination);

    // Sort flights by destination (bubble sort)
    for (int i = 0; i < *flightCount - 1; i++) {
        for (int j = 0; j < *flightCount - i - 1; j++) {
            if (strcasecmp(flights[j].destination, flights[j + 1].destination)
> 0) {
                Flight temp = flights[j];
                flights[j] = flights[j + 1];
                flights[j + 1] = temp;
            }
        }
    }

    // Binary search for flights by destination
    int left = 0;
    int right = *flightCount - 1;
    int mid = -1;
    int found = 0;

    while (left <= right) {
        mid = left + (right - left) / 2;
        int cmp = strcasecmp(flights[mid].destination, destination);

        if (cmp == 0) {
            found = 1;
            break;
        }

        if (cmp < 0) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }

    if (!found) {
        printf("No flights found for destination %s.\n", destination);
        return;
    }

    // Now we've found one matching flight at 'mid'
    // Find all flights with this destination
    int matchingFlights[MAX_FLIGHTS];
    int matchCount = 0;

    // First, add the flight at mid
    matchingFlights[matchCount++] = mid;

    // Check for other matches to the left
    int idx = mid - 1;
    while (idx >= 0 && strcasecmp(flights[idx].destination, destination) == 0)
{
        matchingFlights[matchCount++] = idx;
        idx--;
    }

    // Check for other matches to the right
    idx = mid + 1;
    while (idx < *flightCount && strcasecmp(flights[idx].destination,
destination) == 0) {
        matchingFlights[matchCount++] = idx;
        idx++;
    }

    // Display all matching flights
    for (int i = 0; i < matchCount; i++) {
        int flightIdx = matchingFlights[i];
        printf("Flight Number: %d, Available Business Seats: %d, Available
Economy Seats: %d\n",
               flights[flightIdx].flightNumber,
               flights[flightIdx].availableBusinessSeats,
               flights[flightIdx].availableEconomySeats);
    }

    // Get class type preference
    int classType;
    printf("Select class (1 for Business, 2 for Economy): ");
    scanf("%d", &classType);

    if (classType != 1 && classType != 2) {
        printf("Invalid class type selected.\n");
        return;
    }

    // Use the first matching flight (as per requirements)
    int flightIndex = matchingFlights[0];

    // Check seat availability
    if (classType == 1 && flights[flightIndex].availableBusinessSeats == 0) {
        printf("No available seats in Business class.\n");
        return;
    } else if (classType == 2 && flights[flightIndex].availableEconomySeats ==
0) {
        printf("No available seats in Economy class.\n");
        return;
    }

    // Find first available seat
    int seatNumber = findFirstAvailableSeat(flightIndex, classType);

    if (seatNumber == -1) {
        printf("Error finding an available seat.\n");
        return;
    }

    // Reserve the seat
    if (classType == 1) {
        flights[flightIndex].businessSeats[seatNumber] = 1;
        flights[flightIndex].availableBusinessSeats--;
    } else {
        flights[flightIndex].economySeats[seatNumber] = 1;
        flights[flightIndex].availableEconomySeats--;
    }

    // Create a reservation
    Reservation newReservation;
    strcpy(newReservation.passengerName, customerName);
    newReservation.flightNumber = flights[flightIndex].flightNumber;
    strcpy(newReservation.destination, flights[flightIndex].destination);
    newReservation.seatNumber = seatNumber;
    newReservation.classType = classType;
    newReservation.price = (classType == 1) ?
flights[flightIndex].businessPrice : flights[flightIndex].economyPrice;

    // Add the reservation to the array
    reservations[*reservationCount] = newReservation;
    (*reservationCount)++;

    printf("Reservation successful! Flight %d, Seat Number: %d, Class: %s\n",
           flights[flightIndex].flightNumber,
           seatNumber,
           (classType == 1) ? "Business" : "Economy");
```

```
}

// Cancel a reservation (Customer function) - Optimized with binary search
void cancelReservation(char *customerName) {
    if (*reservationCount == 0) {
        printf("No reservations to cancel.\n");
        return;
    }

    int flightNumber, seatNumber, classType;

    printf("Enter Flight Number: ");
    scanf("%d", &flightNumber);

    // Binary search for flight
    int flightIndex = findFlightByNumber(flightNumber);
    if (flightIndex == -1) {
        printf("Flight %d does not exist.\n", flightNumber);
        return;
    }

    printf("Enter Seat Number: ");
    scanf("%d", &seatNumber);

    printf("Enter Class Type (Business: 1, Economy: 2): ");
    scanf("%d", &classType);

    if (classType != 1 && classType != 2) {
        printf("Invalid class type selected.\n");
        return;
    }

    // Check if the seat is reserved
    if ((classType == 1 && seatNumber >= MAX_BUSINESS_SEATS) ||
        (classType == 2 && seatNumber >= MAX_ECONOMY_SEATS)) {
        printf("Invalid seat number.\n");
        return;
    }

    if ((classType == 1 && flights[flightIndex].businessSeats[seatNumber] == 0)
||
        (classType == 2 && flights[flightIndex].economySeats[seatNumber] == 0))
{
        printf("Seat %d in Flight %d is not reserved. Nothing to be
canceled.\n",
               seatNumber, flightNumber);
        return;
    }

    // Sort reservations first by flightNumber, then by seatNumber
    for (int i = 0; i < *reservationCount - 1; i++) {
        for (int j = 0; j < *reservationCount - i - 1; j++) {
            if (reservations[j].flightNumber > reservations[j + 1].flightNumber
||
                (reservations[j].flightNumber == reservations[j +
1].flightNumber &&
                 reservations[j].seatNumber > reservations[j + 1].seatNumber))
{
                Reservation temp = reservations[j];
                reservations[j] = reservations[j + 1];
                reservations[j + 1] = temp;
            }
        }
    }

    // Binary search for reservation
    int left = 0;
    int right = *reservationCount - 1;
    int reservationIndex = -1;

    while (left <= right) {
        int mid = left + (right - left) / 2;

        if (reservations[mid].flightNumber == flightNumber &&
            reservations[mid].seatNumber == seatNumber &&
            reservations[mid].classType == classType &&
            strcmp(reservations[mid].passengerName, customerName) == 0) {
            reservationIndex = mid;
            break;
        }

        if (reservations[mid].flightNumber < flightNumber ||
            (reservations[mid].flightNumber == flightNumber &&
reservations[mid].seatNumber < seatNumber)) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }

    if (reservationIndex == -1) {
        // Linear search if binary search fails due to name comparison
        for (int i = 0; i < *reservationCount; i++) {
            if (reservations[i].flightNumber == flightNumber &&
                reservations[i].seatNumber == seatNumber &&
                reservations[i].classType == classType &&
                strcmp(reservations[i].passengerName, customerName) == 0) {
                reservationIndex = i;
                break;
            }
        }

        if (reservationIndex == -1) {
            printf("No reservation found for this flight, seat, and class
type.\n");
            return;
        }
    }

    // Mark the seat as available
    if (classType == 1) {
        flights[flightIndex].businessSeats[seatNumber] = 0;
        flights[flightIndex].availableBusinessSeats++;
    } else {
        flights[flightIndex].economySeats[seatNumber] = 0;
        flights[flightIndex].availableEconomySeats++;
    }

    // Remove the reservation (shift all elements after it one position to the
left)
    for (int i = reservationIndex; i < *reservationCount - 1; i++) {
        reservations[i] = reservations[i + 1];
    }
    (*reservationCount)--;

    printf("Reservation for Flight %d and Seat %d has been canceled.\n",
flightNumber, seatNumber);
}
```

```c
// Check total amount spent by a customer (Customer function)
void checkTotalAmountSpent(char *customerName) {
    float totalAmount = 0.0;

    // Sort reservations by passenger name for more efficient searching
    for (int i = 0; i < *reservationCount - 1; i++) {
        for (int j = 0; j < *reservationCount - i - 1; j++) {
            if (strcmp(reservations[j].passengerName, reservations[j +
1].passengerName) > 0) {
                Reservation temp = reservations[j];
                reservations[j] = reservations[j + 1];
                reservations[j + 1] = temp;
            }
        }
    }

    // Find first occurrence of the customer name
    int left = 0;
    int right = *reservationCount - 1;
    int firstIndex = -1;

    while (left <= right) {
        int mid = left + (right - left) / 2;
        int cmp = strcmp(reservations[mid].passengerName, customerName);

        if (cmp == 0) {
            firstIndex = mid;
            right = mid - 1; // Continue searching for the leftmost occurrence
        } else if (cmp < 0) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }

    if (firstIndex != -1) {
        // Process all reservations for this customer
        int i = firstIndex;
        while (i < *reservationCount && strcmp(reservations[i].passengerName,
customerName) == 0) {
            totalAmount += reservations[i].price;
            i++;
        }
    }

    printf("Total amount spent by %s: %.2f\n", customerName, totalAmount);
}

// Find a flight by its flight number - Optimized with binary search
int findFlightByNumber(int flightNumber) {
    // Sort flights by flight number
    for (int i = 1; i < *flightCount; i++) {
        Flight key = flights[i];
        int j = i - 1;

        while (j >= 0 && flights[j].flightNumber > key.flightNumber) {
            flights[j + 1] = flights[j];
            j--;
        }
        flights[j + 1] = key;
    }

    // Binary search
    int left = 0;
    int right = *flightCount - 1;

    while (left <= right) {
        int mid = left + (right - left) / 2;

        if (flights[mid].flightNumber == flightNumber) {
            return mid;
        }

        if (flights[mid].flightNumber < flightNumber) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }

    return -1; // Flight not found
}

// Find the first available seat in a given class for a flight - Using linear
search
int findFirstAvailableSeat(int flightIndex, int classType) {
    if (classType == 1) { // Business class
        for (int i = 0; i < MAX_BUSINESS_SEATS; i++) {
            if (flights[flightIndex].businessSeats[i] == 0) {
                return i;
            }
        }
    } else if (classType == 2) { // Economy class
        for (int i = 0; i < MAX_ECONOMY_SEATS; i++) {
            if (flights[flightIndex].economySeats[i] == 0) {
                return i;
            }
        }
    }
    return -1; // No available seat found
}
```