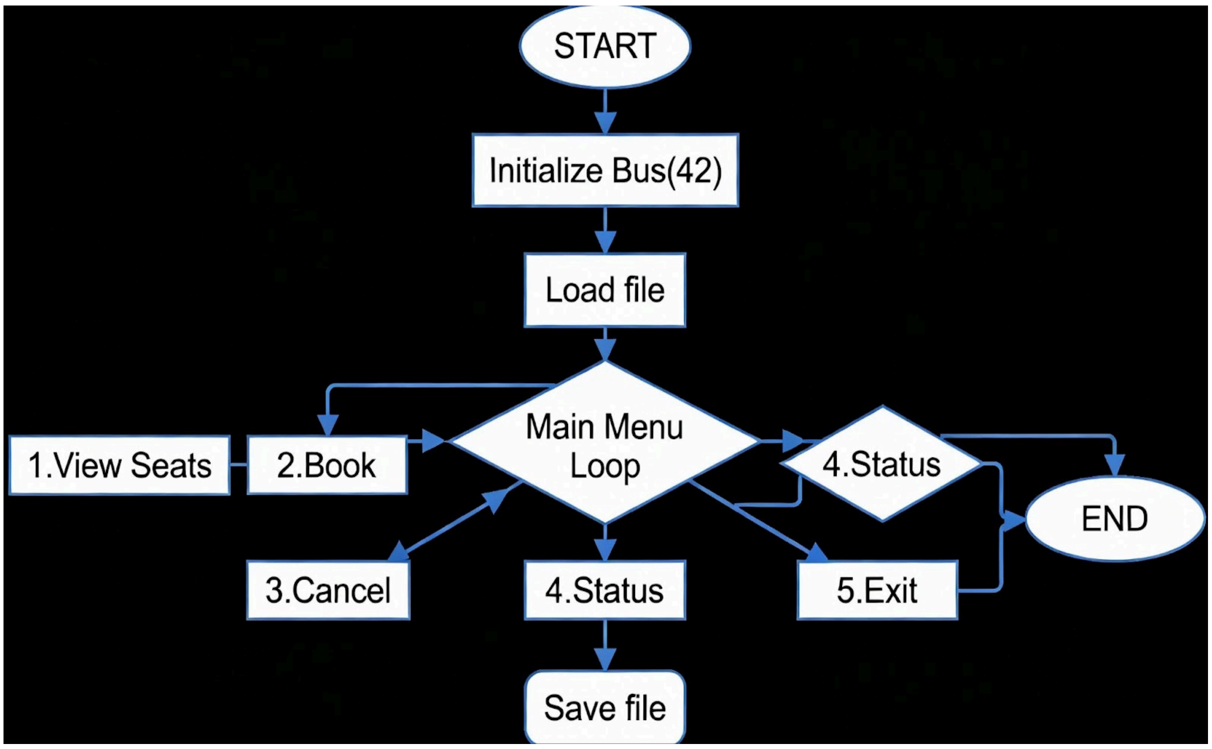
Bus Reservation System

1.Problem definition - The bus reservation system project addresses the need to manage seat bookings on a bus in a simple, accurate, and efficient way using a console-based C program. In many basic setups, reservations, cancellations, and seat status are tracked manually, which can lead to calculation mistakes, double bookings, data loss, and difficulty in knowing current seat availability.

The main problem is to provide a small yet complete system that:

- Stores all seats of a bus with their booking status and passenger name using suitable data structures.
- Allows users to view available seats, book a seat, cancel a booking, and see the full bus status through an interactive menu.
- Maintains data across program runs by saving and loading seat information from a file, so bookings are not lost when the program exits.

2.Flowchart - The flowchart used to approach the problem.



3.Algorithm- Algorithm used to approach the problem:

```
Algorithm: Bus Reservation System

1.Start

2.Define structure BusSeat (seat number, booking status, passenger name).

3.Define structure Bus (dynamic array of seats, capacity).

4.Initialize the bus with 42 seats, allocate memory, set all seats as empty.

5.Load saved data from file if available.

6.Display menu: View Seats, Book Seat, Cancel Seat, View Status, Save & Exit.

7.Repeat until user selects Exit:

  >If View Seats: show all unbooked seats and count.

  >If Book Seat: take seat number and name, validate, then mark as booked.

  >If Cancel Seat: check if booked, then clear passenger name and mark empty.

  >If View Status: show all seats with booking info.

  >If invalid input: display error message.

8.Save bus data to file.

9.Free memory.
```

4.Problems faced by the group-

During development of the bus reservation system, we encountered several technical challenges related to C programming fundamentals, particularly with dynamic memory management and file I/O operations.

Memory Allocation and Pointer Handling Issues:

- Initial difficulties with malloc() for the dynamic seat array in initializeBus(), including handling allocation failures and ensuring proper pointer initialization to avoid segmentation faults.
- Ensuring pointer arithmetic in bookSeat() and cancelSeat() correctly accessed bus->seats[seatNo - 1] without array bounds violations.

File Handling and Data Persistence Challenges:

- Complexities in loadBusFromFile() where fscanf() parsing mismatched or corrupted file data led to incomplete seat loading, requiring fallback to default empty states.
- Synchronization issues between saveBusToFile() and loadBusFromFile(), such as capacity mismatches causing allocation failures or data loss across program restarts.

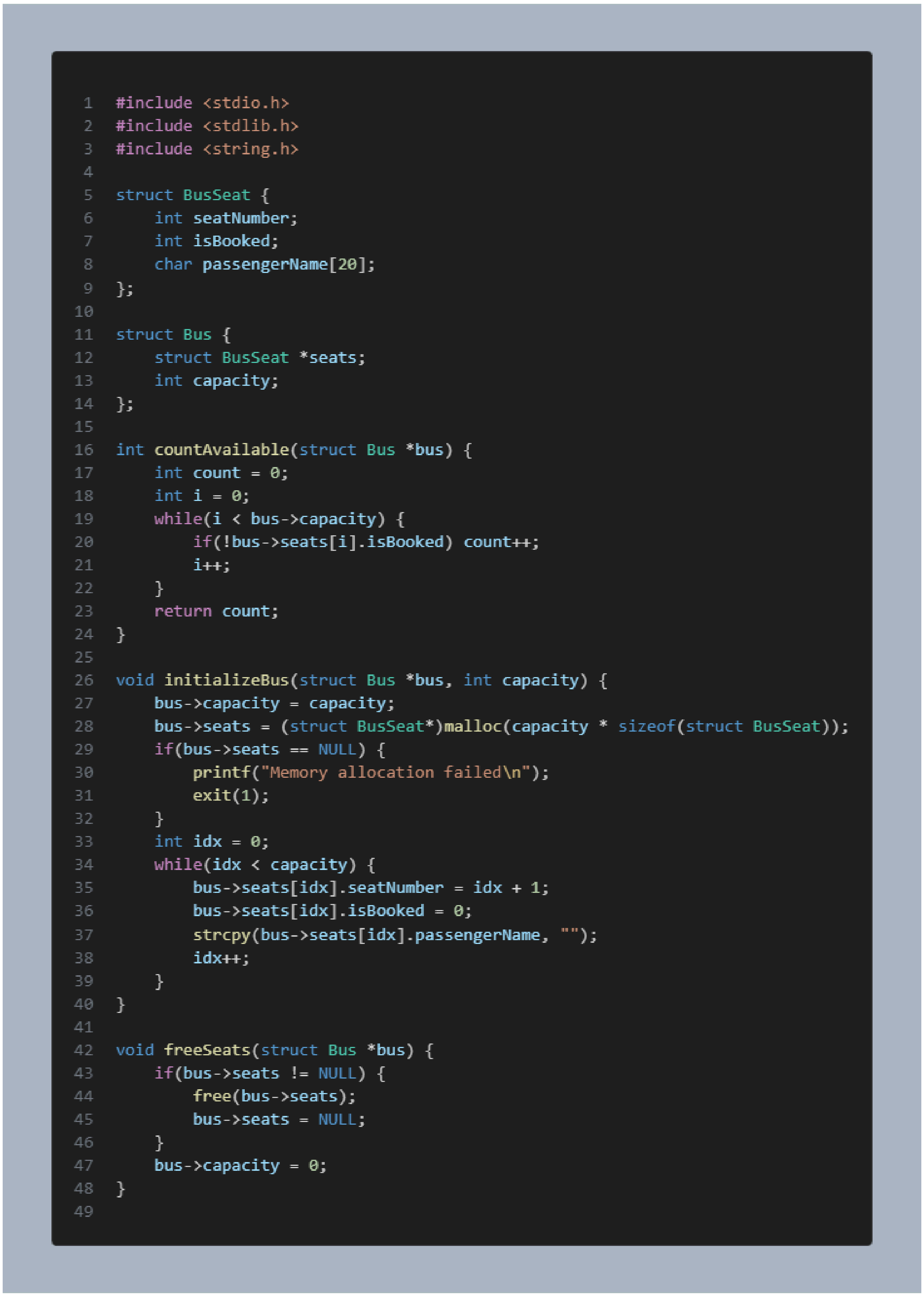
Input Buffer Management:

- Persistent problems with leftover newline characters after scanf("%d", &choice) causing fgets() in booking to read empty strings, resolved using while(getchar() != '\n') but initially led to failed bookings.

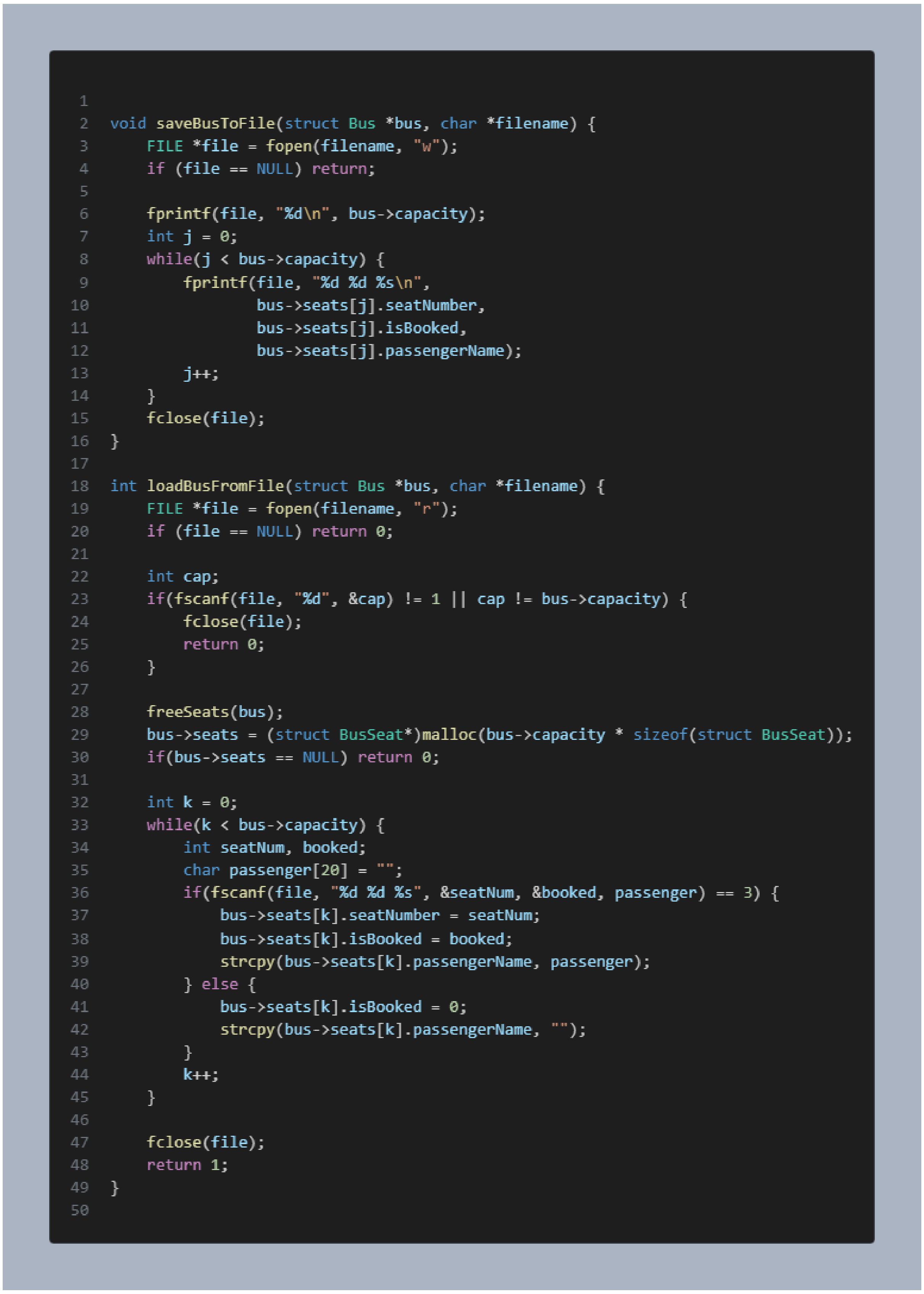
String Manipulation Errors:

- Buffer overflow risks and truncation in strcpy() for passengerName, especially with fgets() and strcspn() to strip newlines, resulting in garbled passenger names during display.

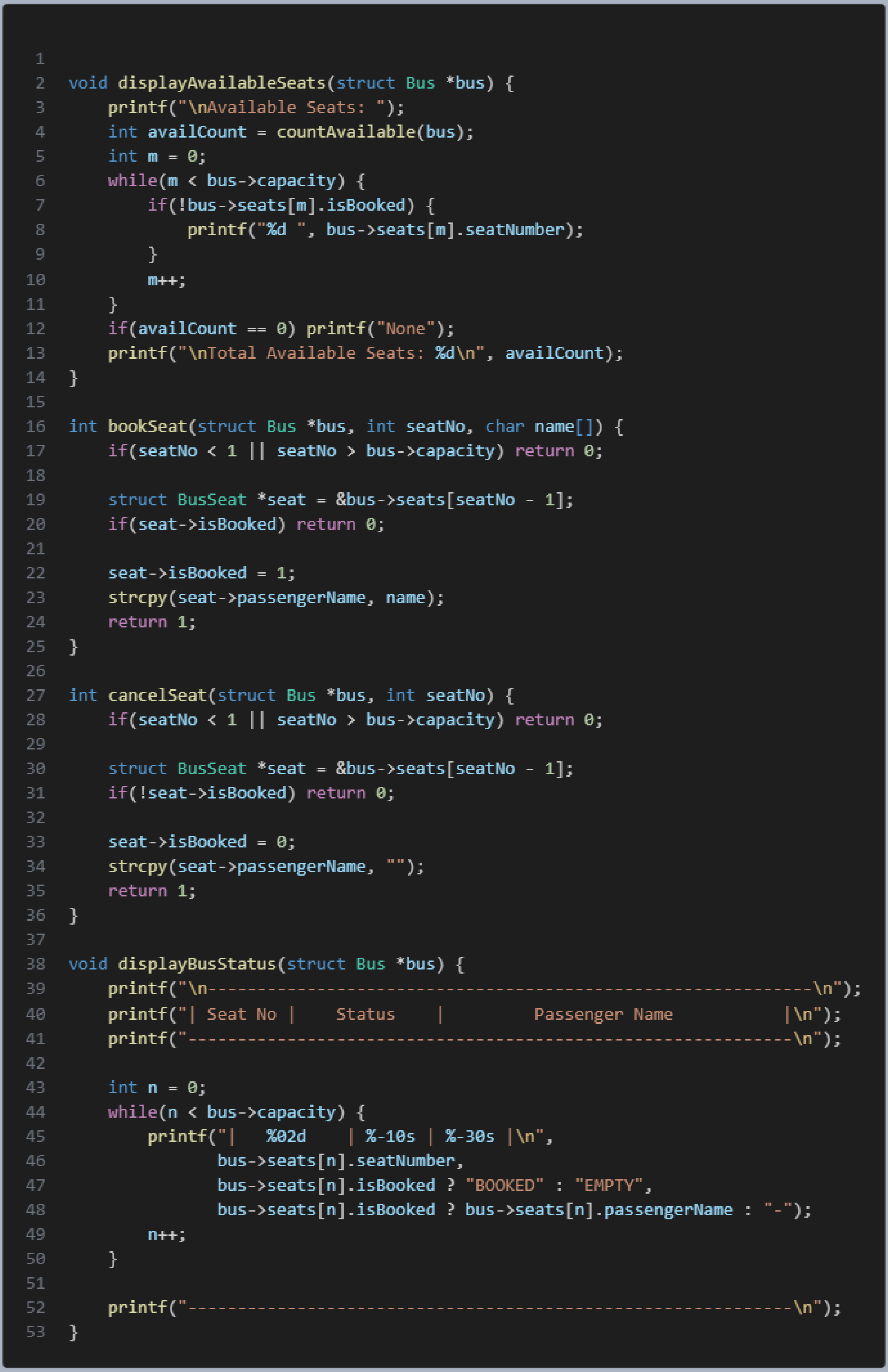
5.Code-



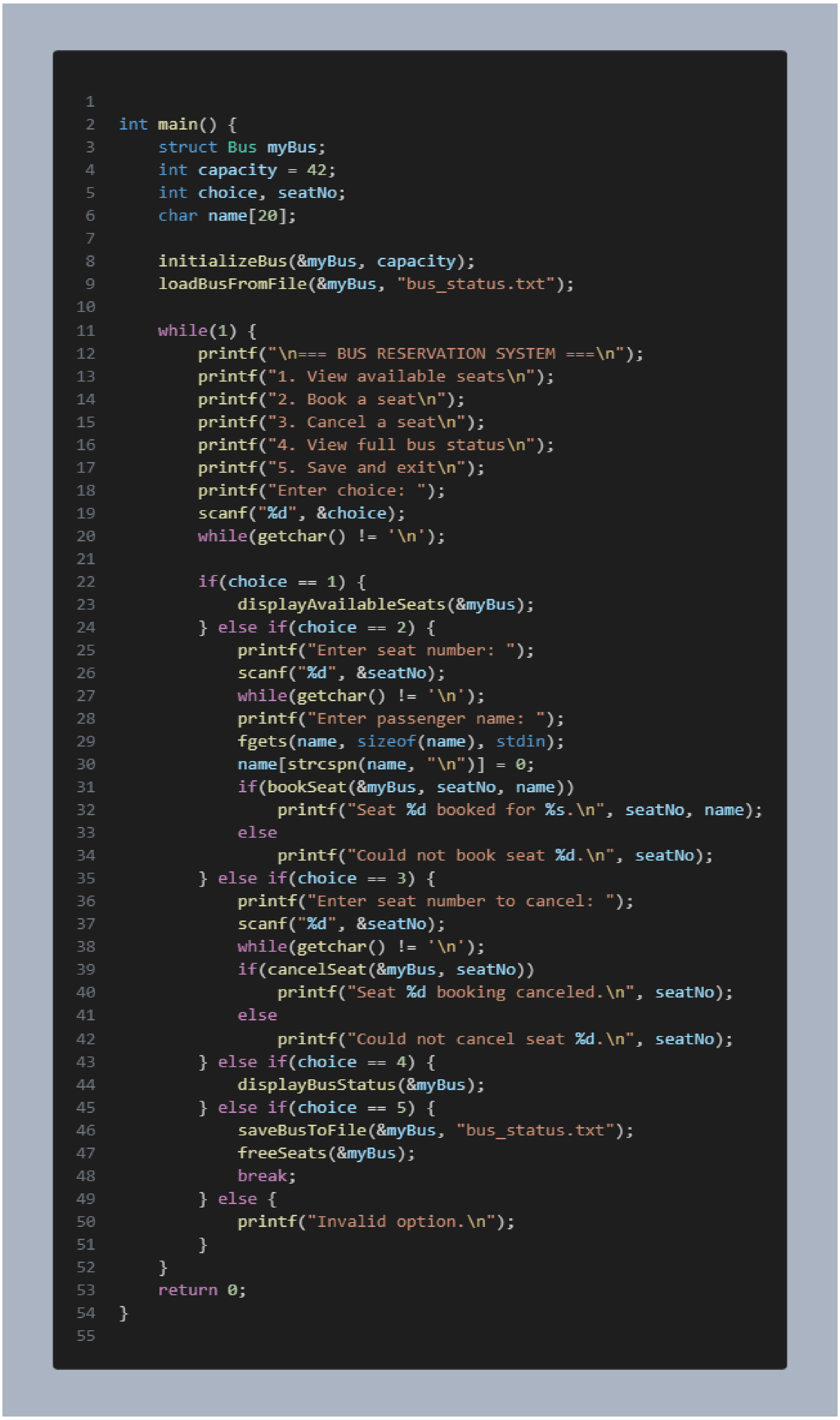
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct BusSeat {
    int seatNumber;
    int isBooked;
    char passengerName[20];
};

struct Bus {
    struct BusSeat *seats;
    int capacity;
};

int countAvailable(struct Bus *bus) {
    int count = 0;
    int i = 0;
    while(i < bus->capacity) {
        if(!bus->seats[i].isBooked) count++;
        i++;
    }
    return count;
}

void initializeBus(struct Bus *bus, int capacity) {
    bus->capacity = capacity;
    bus->seats = (struct BusSeat*)malloc(capacity * sizeof(struct BusSeat));
    if(bus->seats == NULL) {
        printf("Memory allocation failed\n");
        exit(1);
    }
    int idx = 0;
    while(idx < capacity) {
        bus->seats[idx].seatNumber = idx + 1;
        bus->seats[idx].isBooked = 0;
        strcpy(bus->seats[idx].passengerName, "");
        idx++;
    }
}

void freeSeats(struct Bus *bus) {
    if(bus->seats != NULL) {
        free(bus->seats);
        bus->seats = NULL;
    }
    bus->capacity = 0;
}

```



```c

void saveBusToFile(struct Bus *bus, char *filename) {
    FILE *file = fopen(filename, "w");
    if (file == NULL) return;

    fprintf(file, "%d\n", bus->capacity);
    int j = 0;
    while(j < bus->capacity) {
        fprintf(file, "%d %d %s\n",
                bus->seats[j].seatNumber,
                bus->seats[j].isBooked,
                bus->seats[j].passengerName);
        j++;
    }
    fclose(file);
}

int loadBusFromFile(struct Bus *bus, char *filename) {
    FILE *file = fopen(filename, "r");
    if (file == NULL) return 0;

    int cap;
    if(fscanf(file, "%d", &cap) != 1 || cap != bus->capacity) {
        fclose(file);
        return 0;
    }

    freeSeats(bus);
    bus->seats = (struct BusSeat*)malloc(bus->capacity * sizeof(struct BusSeat));
    if(bus->seats == NULL) return 0;

    int k = 0;
    while(k < bus->capacity) {
        int seatNum, booked;
        char passenger[20] = "";
        if(fscanf(file, "%d %d %s", &seatNum, &booked, passenger) == 3) {
            bus->seats[k].seatNumber = seatNum;
            bus->seats[k].isBooked = booked;
            strcpy(bus->seats[k].passengerName, passenger);
        } else {
            bus->seats[k].isBooked = 0;
            strcpy(bus->seats[k].passengerName, "");
        }
        k++;
    }

    fclose(file);
    return 1;
}

```

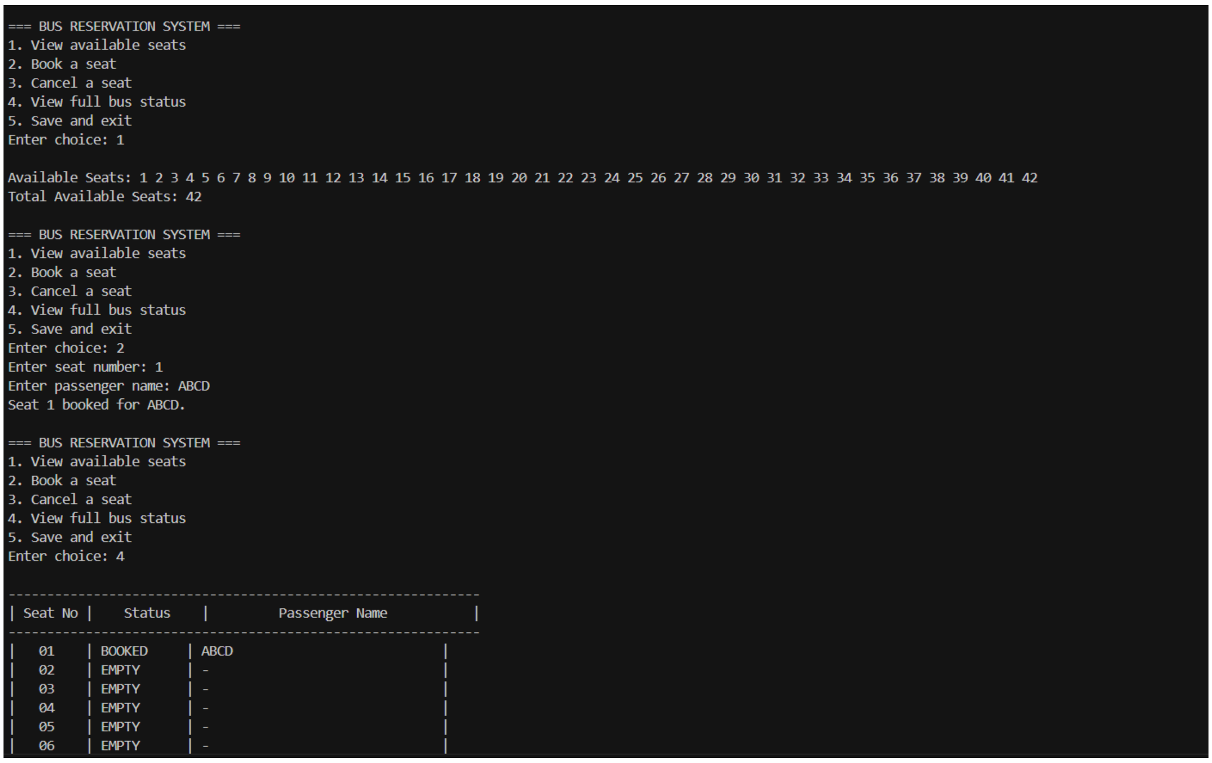


```c

void displayAvailableSeats(struct Bus *bus) {
    printf("\nAvailable Seats: ");
    int availCount = countAvailable(bus);
    int m = 0;
    while(m < bus->capacity) {
        if(!bus->seats[m].isBooked) {
            printf("%d ", bus->seats[m].seatNumber);
        }
        m++;
    }
    if(availCount == 0) printf("None");
    printf("\nTotal Available Seats: %d\n", availCount);
}

int bookSeat(struct Bus *bus, int seatNo, char name[]) {
    if(seatNo < 1 || seatNo > bus->capacity) return 0;

    struct BusSeat *seat = &bus->seats[seatNo - 1];
    if(seat->isBooked) return 0;

    seat->isBooked = 1;
    strcpy(seat->passengerName, name);
    return 1;
}

int cancelSeat(struct Bus *bus, int seatNo) {
    if(seatNo < 1 || seatNo > bus->capacity) return 0;

    struct BusSeat *seat = &bus->seats[seatNo - 1];
    if(!seat->isBooked) return 0;

    seat->isBooked = 0;
    strcpy(seat->passengerName, "");
    return 1;
}

void displayBusStatus(struct Bus *bus) {
    printf("\n-------------------------------------------------------------\n");
    printf("| Seat No |    Status    |         Passenger Name           |\n");
    printf("-------------------------------------------------------------\n");

    int n = 0;
    while(n < bus->capacity) {
        printf("|   %02d    | %-10s | %-30s |\n",
               bus->seats[n].seatNumber,
               bus->seats[n].isBooked ? "BOOKED" : "EMPTY",
               bus->seats[n].isBooked ? bus->seats[n].passengerName : "-");
        n++;
    }

    printf("-------------------------------------------------------------\n");
}
```



```c

int main() {
    struct Bus myBus;
    int capacity = 42;
    int choice, seatNo;
    char name[20];

    initializeBus(&myBus, capacity);
    loadBusFromFile(&myBus, "bus_status.txt");

    while(1) {
        printf("\n=== BUS RESERVATION SYSTEM ===\n");
        printf("1. View available seats\n");
        printf("2. Book a seat\n");
        printf("3. Cancel a seat\n");
        printf("4. View full bus status\n");
        printf("5. Save and exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);
        while(getchar() != '\n');

        if(choice == 1) {
            displayAvailableSeats(&myBus);
        } else if(choice == 2) {
            printf("Enter seat number: ");
            scanf("%d", &seatNo);
            while(getchar() != '\n');
            printf("Enter passenger name: ");
            fgets(name, sizeof(name), stdin);
            name[strcspn(name, "\n")] = 0;
            if(bookSeat(&myBus, seatNo, name))
                printf("Seat %d booked for %s.\n", seatNo, name);
            else
                printf("Could not book seat %d.\n", seatNo);
        } else if(choice == 3) {
            printf("Enter seat number to cancel: ");
            scanf("%d", &seatNo);
            while(getchar() != '\n');
            if(cancelSeat(&myBus, seatNo))
                printf("Seat %d booking canceled.\n", seatNo);
            else
                printf("Could not cancel seat %d.\n", seatNo);
        } else if(choice == 4) {
            displayBusStatus(&myBus);
        } else if(choice == 5) {
            saveBusToFile(&myBus, "bus_status.txt");
            freeSeats(&myBus);
            break;
        } else {
            printf("Invalid option.\n");
        }
    }
    return 0;
}

```

6.Output-



```
=== BUS RESERVATION SYSTEM ===
1. View available seats
2. Book a seat
3. Cancel a seat
4. View full bus status
5. Save and exit
Enter choice: 1

Available Seats: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42
Total Available Seats: 42

=== BUS RESERVATION SYSTEM ===
1. View available seats
2. Book a seat
3. Cancel a seat
4. View full bus status
5. Save and exit
Enter choice: 2
Enter seat number: 1
Enter passenger name: ABCD
Seat 1 booked for ABCD.

=== BUS RESERVATION SYSTEM ===
1. View available seats
2. Book a seat
3. Cancel a seat
4. View full bus status
5. Save and exit
Enter choice: 4

-------------------------------------------------------------
| Seat No |    Status    |         Passenger Name          |
-------------------------------------------------------------
|   01    | BOOKED      | ABCD                            |
|   02    | EMPTY       | -                               |
|   03    | EMPTY       | -                               |
|   04    | EMPTY       | -                               |
|   05    | EMPTY       | -                               |
|   06    | EMPTY       | -                               |
```