Railway Reservation System: Simplifying Ticket Booking

Explore the world of seamless ticket booking with our Railway Reservation System. Say goodbye to long queues and enjoy hassle-free travel!

M **Nikitha T - PES2UG22CS362**

M **Nikita Anup - PES2UG22CS361**

M **Mrunal Anandache - PES2UG22CS323**

Synopsis

A robust railway reservation system program implemented in C language using the singly linked list data structure. This project aims to visualise the linked list and its functionalities in the real world.

Challenges in the current system:

1 Inefficiencies

The existing ticketing system lacks automation, causing delays and errors in ticket generation and passenger information management.

2 Limited Access

Many people, especially in remote areas, cannot easily access ticketing centers, leading to inconvenience and restricted travel options.

Technical Aspects

Programming Language	C
Data Structure	Singly linked list

ADT:

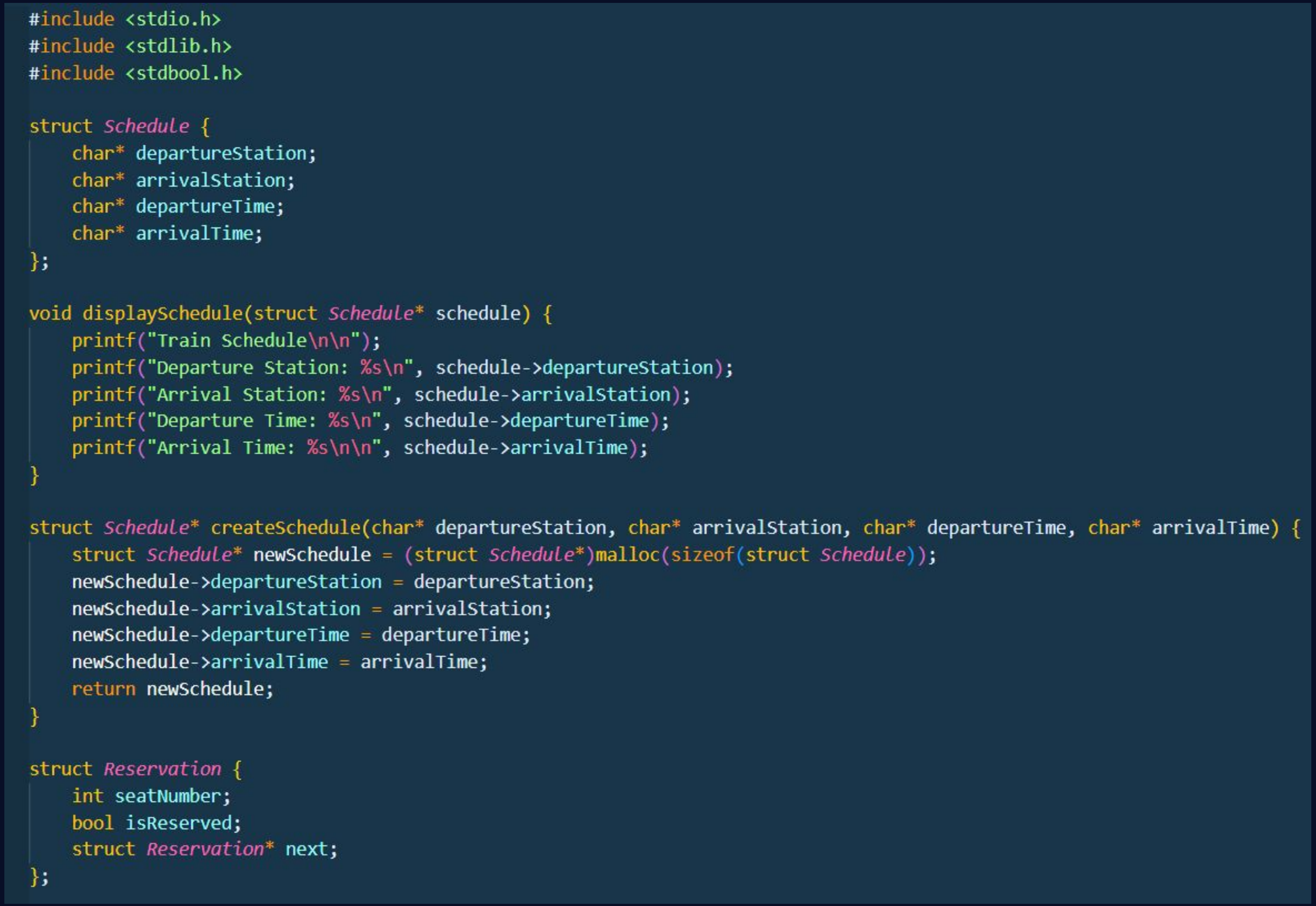
- Linked List can be defined as collection of objects called **nodes** that are randomly stored in the memory.
- A node contains two fields i.e. data stored at that particular address and the pointer which contains the address of the next node in the memory.
- The last node of the list contains pointer to the null.

Solution in summary:

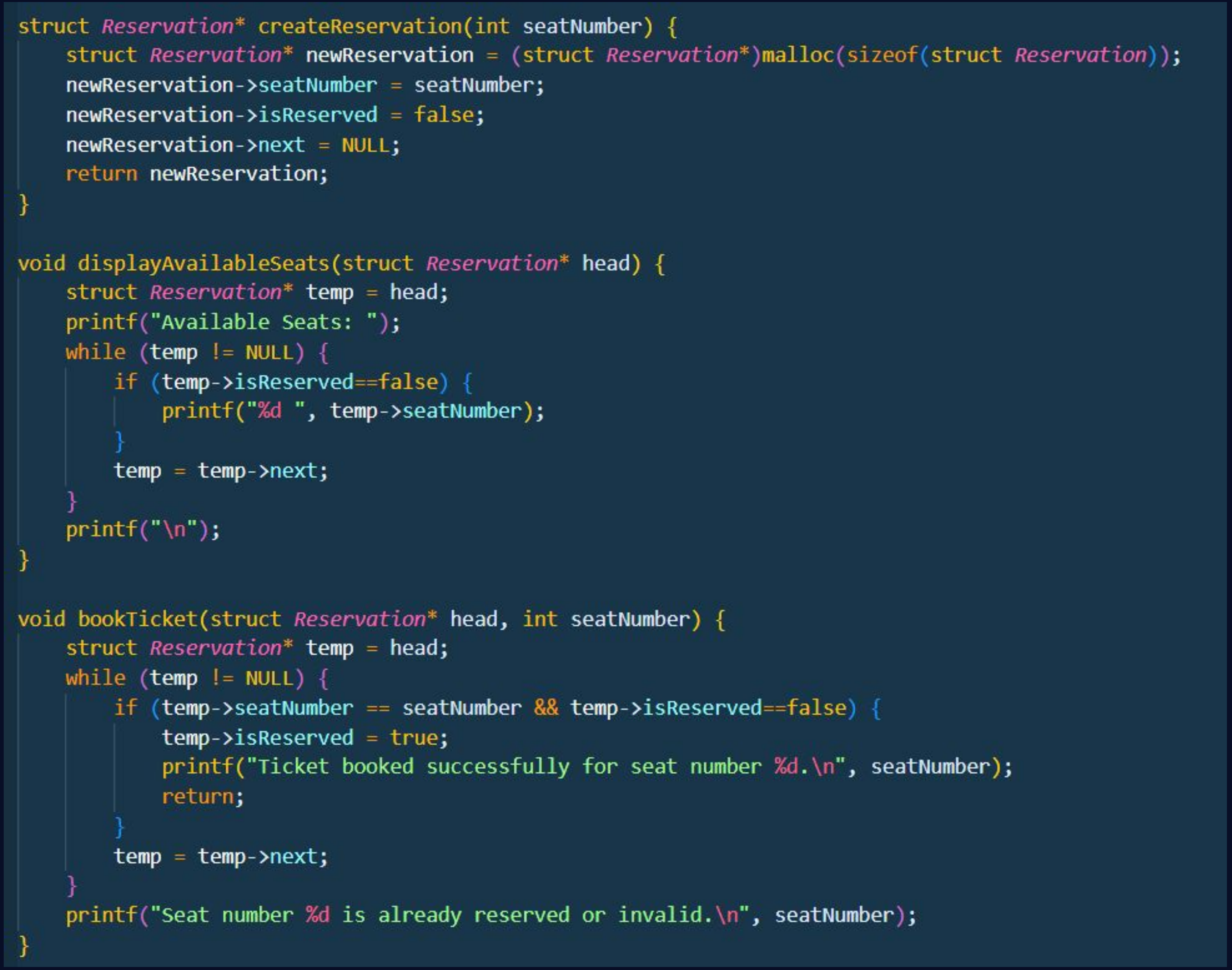
Operations performed are:

1) Insertion of nodes (at front) - To initialise (reserve) seats in the train.
2) Traversal of nodes - to facilitate booking and cancellation of a reservation.
3) Display data - to display available seats and current schedule of a train.
4) Free linked list - to free up memory after exit.

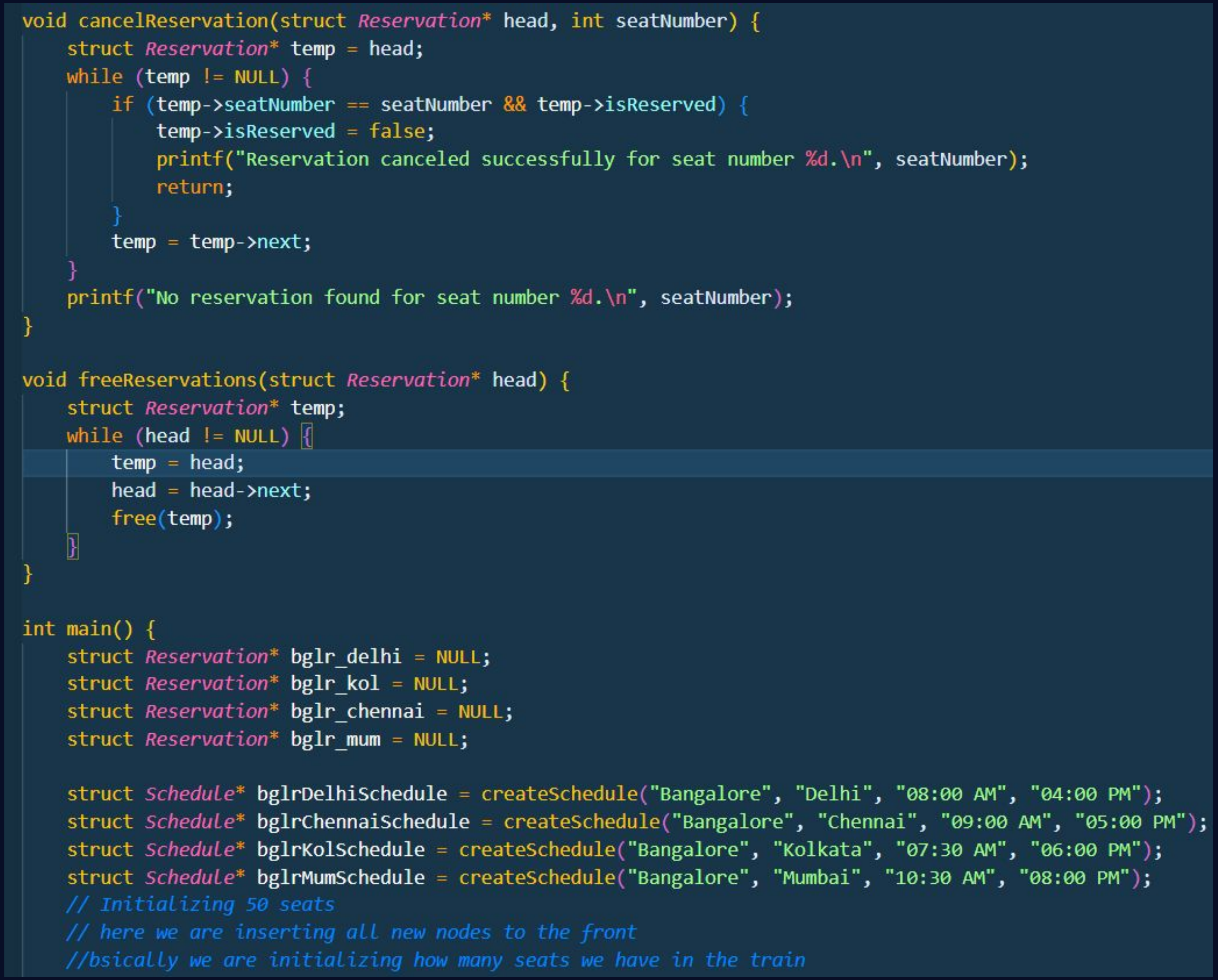
CODE



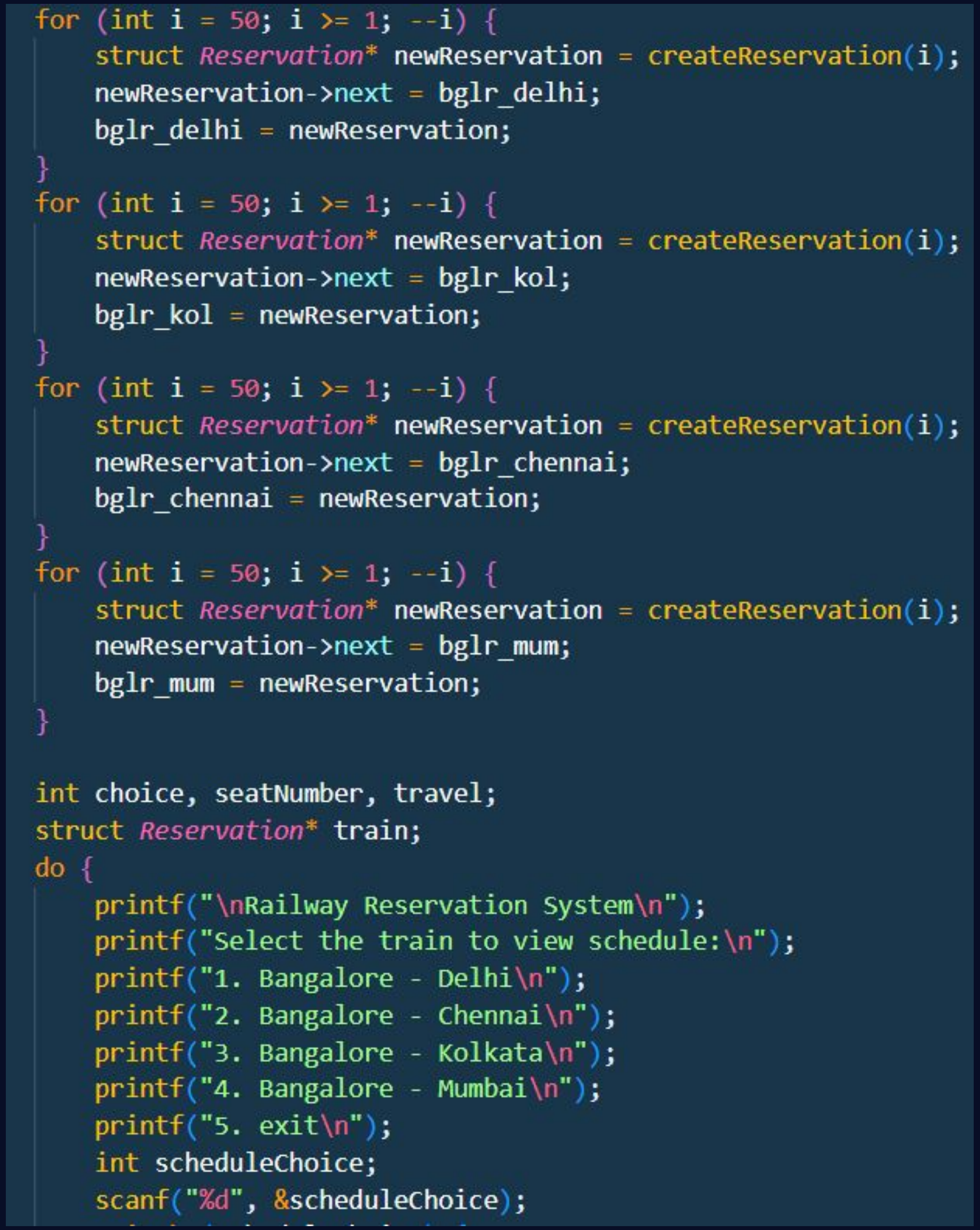
```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

struct Schedule {
    char* departureStation;
    char* arrivalStation;
    char* departureTime;
    char* arrivalTime;
};

void displaySchedule(struct Schedule* schedule) {
    printf("Train Schedule\n\n");
    printf("Departure Station: %s\n", schedule->departureStation);
    printf("Arrival Station: %s\n", schedule->arrivalStation);
    printf("Departure Time: %s\n", schedule->departureTime);
    printf("Arrival Time: %s\n\n", schedule->arrivalTime);
}

struct Schedule* createSchedule(char* departureStation, char* arrivalStation, char* departureTime, char* arrivalTime) {
    struct Schedule* newSchedule = (struct Schedule*)malloc(sizeof(struct Schedule));
    newSchedule->departureStation = departureStation;
    newSchedule->arrivalStation = arrivalStation;
    newSchedule->departureTime = departureTime;
    newSchedule->arrivalTime = arrivalTime;
    return newSchedule;
}

struct Reservation {
    int seatNumber;
    bool isReserved;
    struct Reservation* next;
};
```



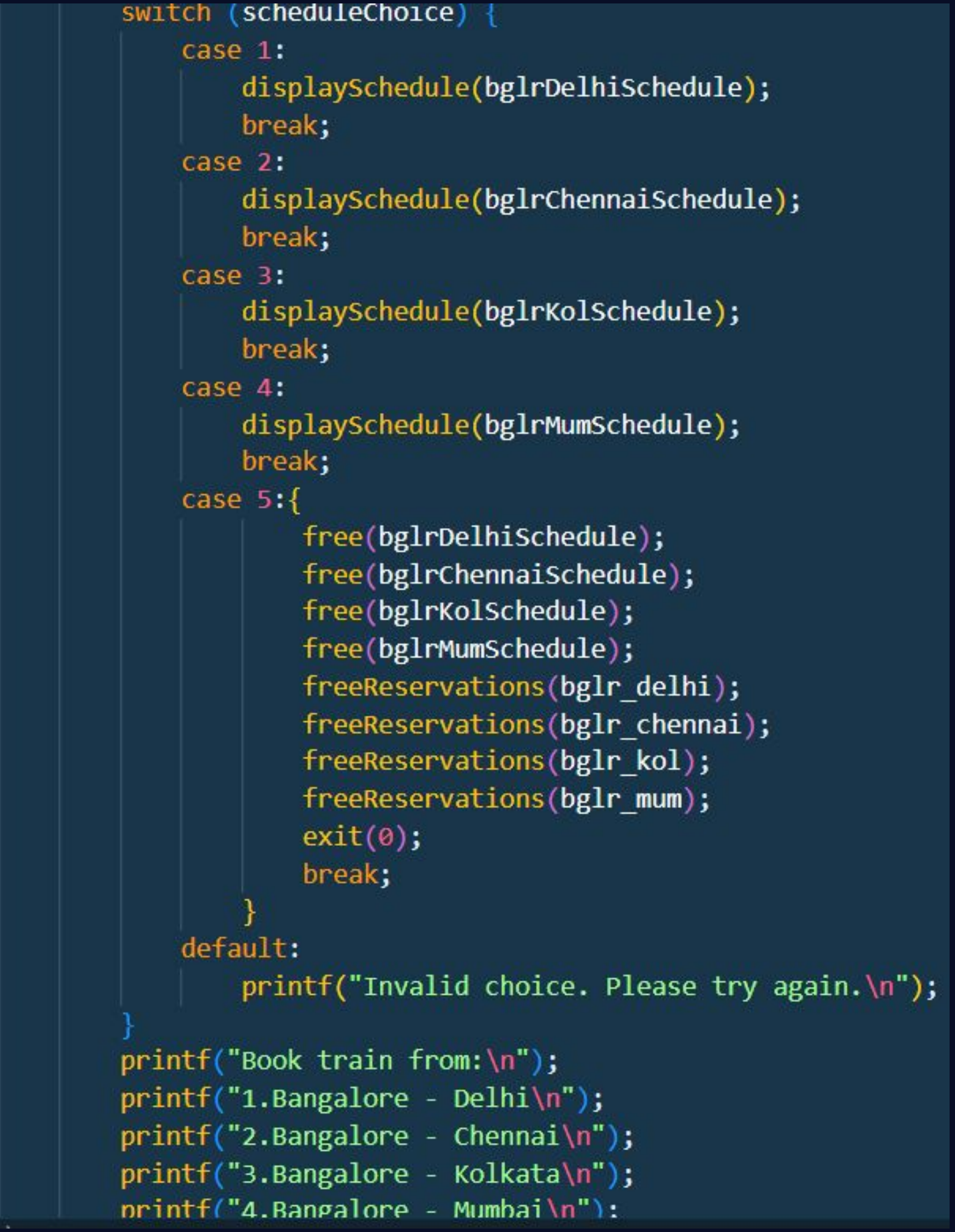
```c
struct Reservation* createReservation(int seatNumber) {
    struct Reservation* newReservation = (struct Reservation*)malloc(sizeof(struct Reservation));
    newReservation->seatNumber = seatNumber;
    newReservation->isReserved = false;
    newReservation->next = NULL;
    return newReservation;
}

void displayAvailableSeats(struct Reservation* head) {
    struct Reservation* temp = head;
    printf("Available Seats: ");
    while (temp != NULL) {
        if (temp->isReserved==false) {
            printf("%d ", temp->seatNumber);
        }
        temp = temp->next;
    }
    printf("\n");
}

void bookTicket(struct Reservation* head, int seatNumber) {
    struct Reservation* temp = head;
    while (temp != NULL) {
        if (temp->seatNumber == seatNumber && temp->isReserved==false) {
            temp->isReserved = true;
            printf("Ticket booked successfully for seat number %d.\n", seatNumber);
            return;
        }
        temp = temp->next;
    }
    printf("Seat number %d is already reserved or invalid.\n", seatNumber);
}
```



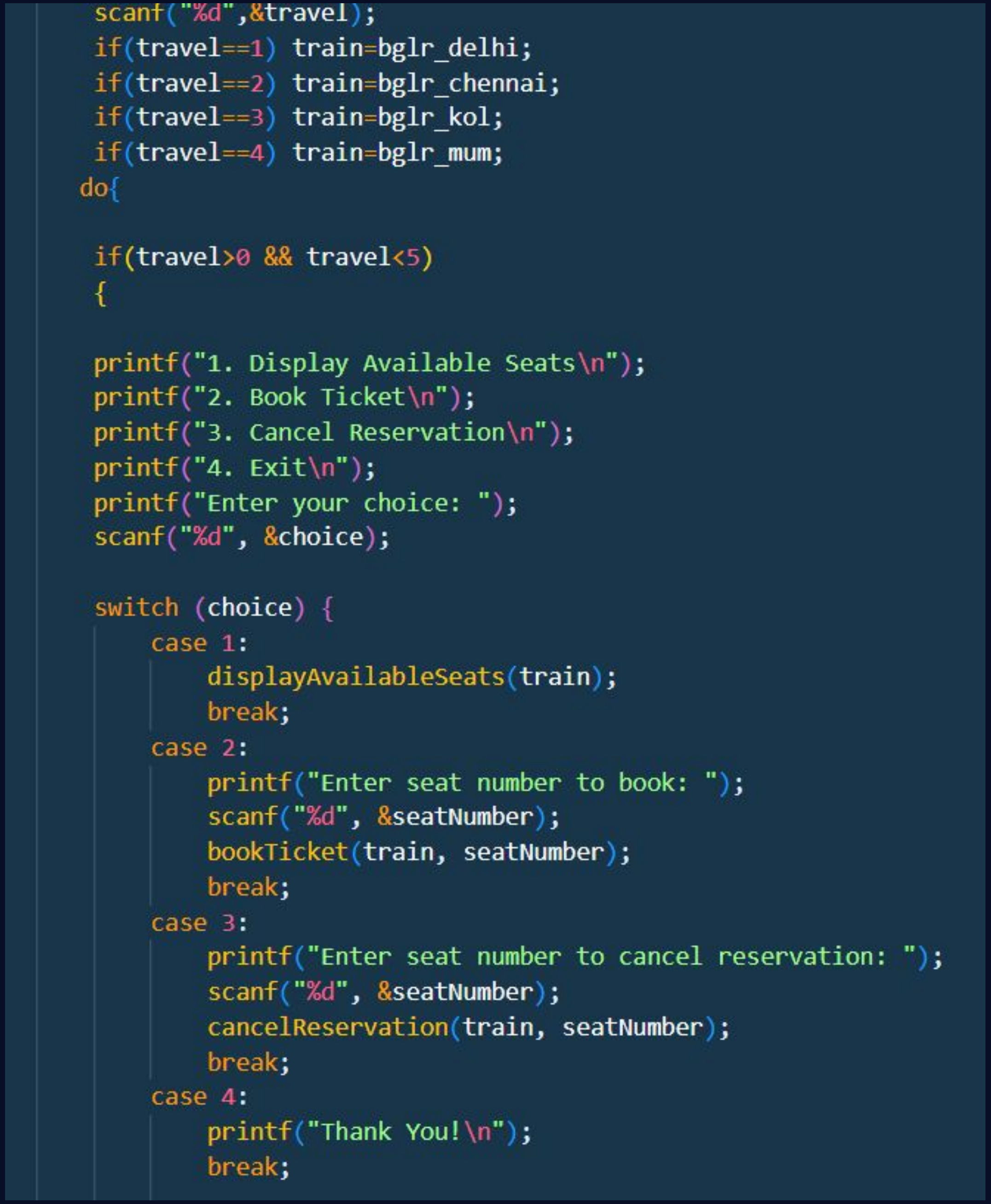
```c
void cancelReservation(struct Reservation* head, int seatNumber) {
    struct Reservation* temp = head;
    while (temp != NULL) {
        if (temp->seatNumber == seatNumber && temp->isReserved) {
            temp->isReserved = false;
            printf("Reservation canceled successfully for seat number %d.\n", seatNumber);
            return;
        }
        temp = temp->next;
    }
    printf("No reservation found for seat number %d.\n", seatNumber);
}

void freeReservations(struct Reservation* head) {
    struct Reservation* temp;
    while (head != NULL) {
        temp = head;
        head = head->next;
        free(temp);
    }
}

int main() {
    struct Reservation* bglr_delhi = NULL;
    struct Reservation* bglr_kol = NULL;
    struct Reservation* bglr_chennai = NULL;
    struct Reservation* bglr_mum = NULL;

    struct Schedule* bglrDelhiSchedule = createSchedule("Bangalore", "Delhi", "08:00 AM", "04:00 PM");
    struct Schedule* bglrChennaiSchedule = createSchedule("Bangalore", "Chennai", "09:00 AM", "05:00 PM");
    struct Schedule* bglrKolSchedule = createSchedule("Bangalore", "Kolkata", "07:30 AM", "06:00 PM");
    struct Schedule* bglrMumSchedule = createSchedule("Bangalore", "Mumbai", "10:30 AM", "08:00 PM");
    // Initializing 50 seats
    // here we are inserting all new nodes to the front
    //bsically we are initializing how many seats we have in the train
```



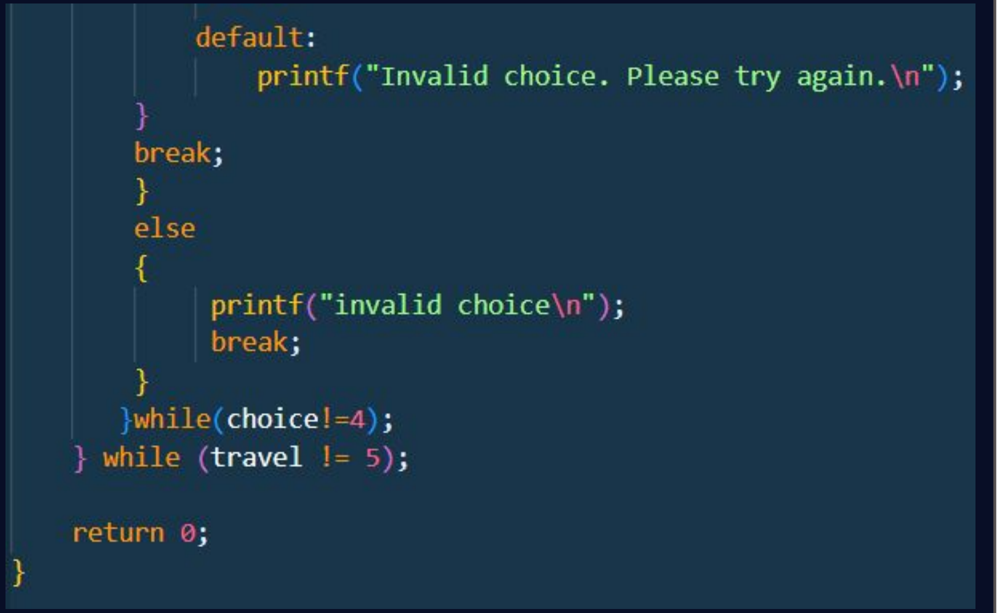
```c
for (int i = 50; i >= 1; --i) {
    struct Reservation* newReservation = createReservation(i);
    newReservation->next = bglr_delhi;
    bglr_delhi = newReservation;
}
for (int i = 50; i >= 1; --i) {
    struct Reservation* newReservation = createReservation(i);
    newReservation->next = bglr_kol;
    bglr_kol = newReservation;
}
for (int i = 50; i >= 1; --i) {
    struct Reservation* newReservation = createReservation(i);
    newReservation->next = bglr_chennai;
    bglr_chennai = newReservation;
}
for (int i = 50; i >= 1; --i) {
    struct Reservation* newReservation = createReservation(i);
    newReservation->next = bglr_mum;
    bglr_mum = newReservation;
}

int choice, seatNumber, travel;
struct Reservation* train;
do {
    printf("\nRailway Reservation System\n");
    printf("Select the train to view schedule:\n");
    printf("1. Bangalore - Delhi\n");
    printf("2. Bangalore - Chennai\n");
    printf("3. Bangalore - Kolkata\n");
    printf("4. Bangalore - Mumbai\n");
    printf("5. exit\n");
    int scheduleChoice;
    scanf("%d", &scheduleChoice);
```



```
    switch (scheduleChoice) {
        case 1:
            displaySchedule(bglrDelhiSchedule);
            break;
        case 2:
            displaySchedule(bglrChennaiSchedule);
            break;
        case 3:
            displaySchedule(bglrKolSchedule);
            break;
        case 4:
            displaySchedule(bglrMumSchedule);
            break;
        case 5:{
                free(bglrDelhiSchedule);
                free(bglrChennaiSchedule);
                free(bglrKolSchedule);
                free(bglrMumSchedule);
                freeReservations(bglr_delhi);
                freeReservations(bglr_chennai);
                freeReservations(bglr_kol);
                freeReservations(bglr_mum);
                exit(0);
                break;
            }
        default:
            printf("Invalid choice. Please try again.\n");
    }
    printf("Book train from:\n");
    printf("1.Bangalore - Delhi\n");
    printf("2.Bangalore - Chennai\n");
    printf("3.Bangalore - Kolkata\n");
    printf("4.Bangalore - Mumbai\n");
```



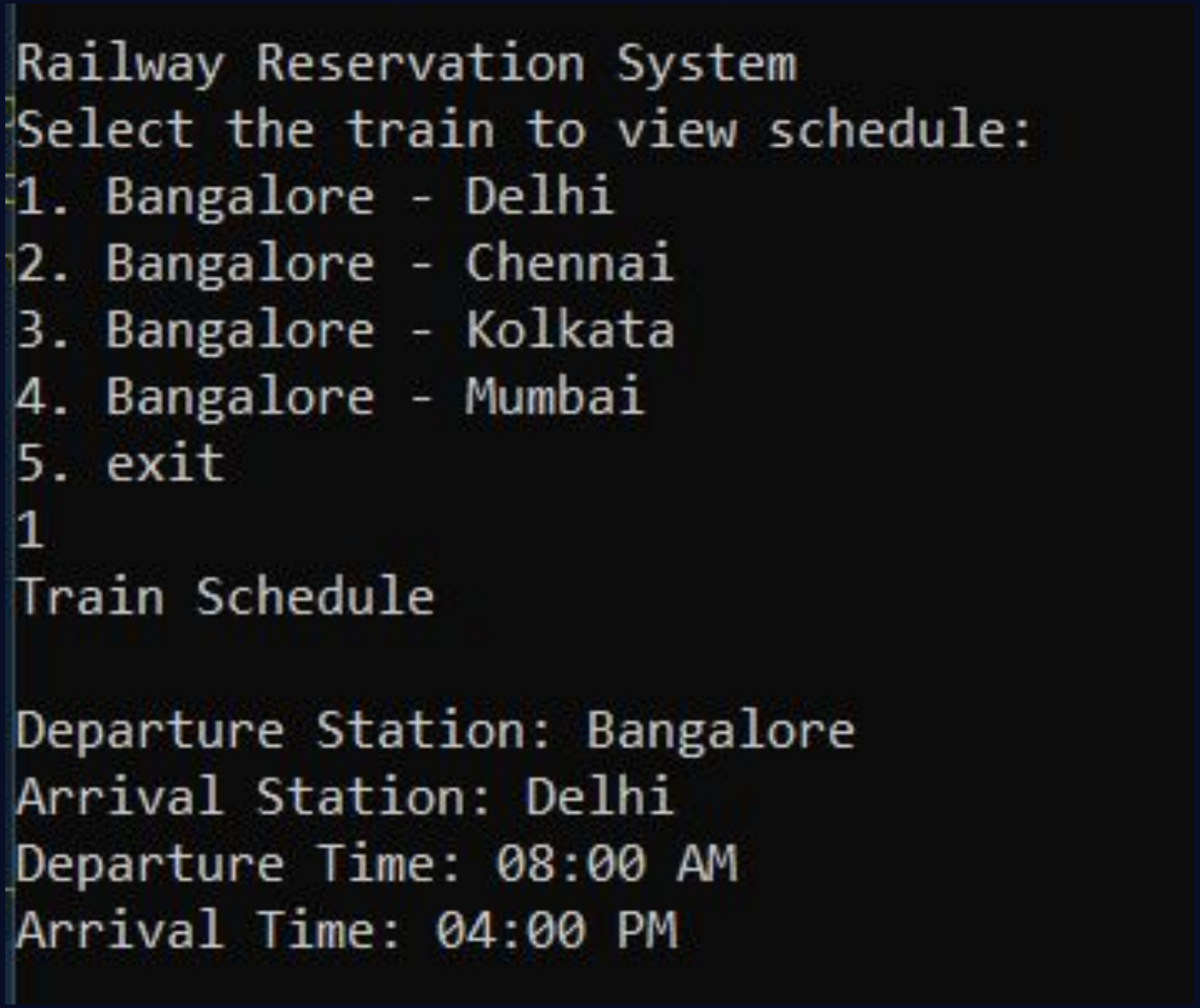
```
    scanf("%d",&travel);
    if(travel==1) train=bglr_delhi;
    if(travel==2) train=bglr_chennai;
    if(travel==3) train=bglr_kol;
    if(travel==4) train=bglr_mum;
   do{

    if(travel>0 && travel<5)
    {

    printf("1. Display Available Seats\n");
    printf("2. Book Ticket\n");
    printf("3. Cancel Reservation\n");
    printf("4. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            displayAvailableSeats(train);
            break;
        case 2:
            printf("Enter seat number to book: ");
            scanf("%d", &seatNumber);
            bookTicket(train, seatNumber);
            break;
        case 3:
            printf("Enter seat number to cancel reservation: ");
            scanf("%d", &seatNumber);
            cancelReservation(train, seatNumber);
            break;
        case 4:
            printf("Thank You!\n");
            break;
```



```
            default:
                printf("Invalid choice. Please try again.\n");
        }
        break;
        }
        else
        {
             printf("invalid choice\n");
             break;
        }
       }while(choice!=4);
    } while (travel != 5);

    return 0;
}
```

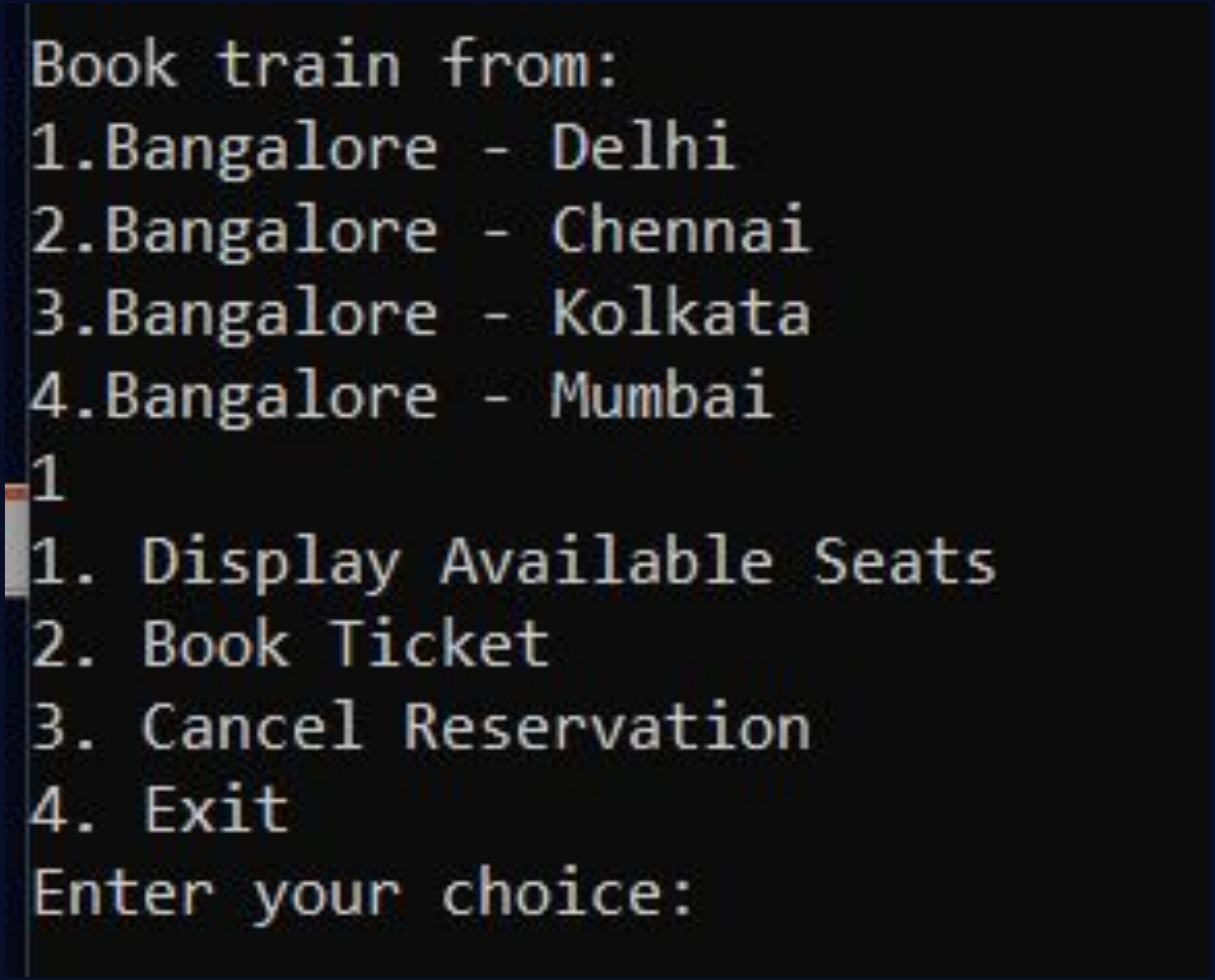
Outputs

i) View Train Schedule



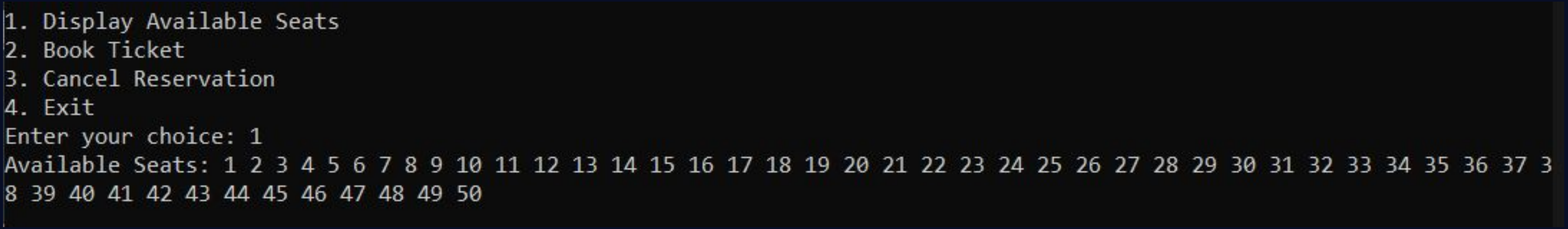
```
Railway Reservation System
Select the train to view schedule:
1. Bangalore - Delhi
2. Bangalore - Chennai
3. Bangalore - Kolkata
4. Bangalore - Mumbai
5. exit
1
Train Schedule

Departure Station: Bangalore
Arrival Station: Delhi
Departure Time: 08:00 AM
Arrival Time: 04:00 PM
```

ii) Booking train



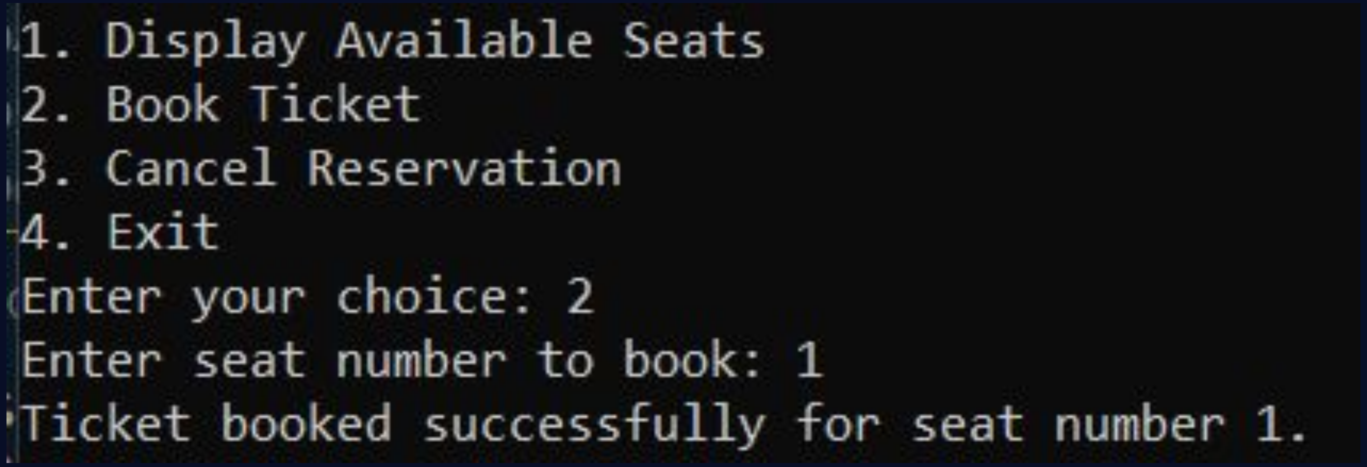
```
Book train from:
1.Bangalore - Delhi
2.Bangalore - Chennai
3.Bangalore - Kolkata
4.Bangalore - Mumbai
1
1. Display Available Seats
2. Book Ticket
3. Cancel Reservation
4. Exit
Enter your choice:
```

iii) Available Seats



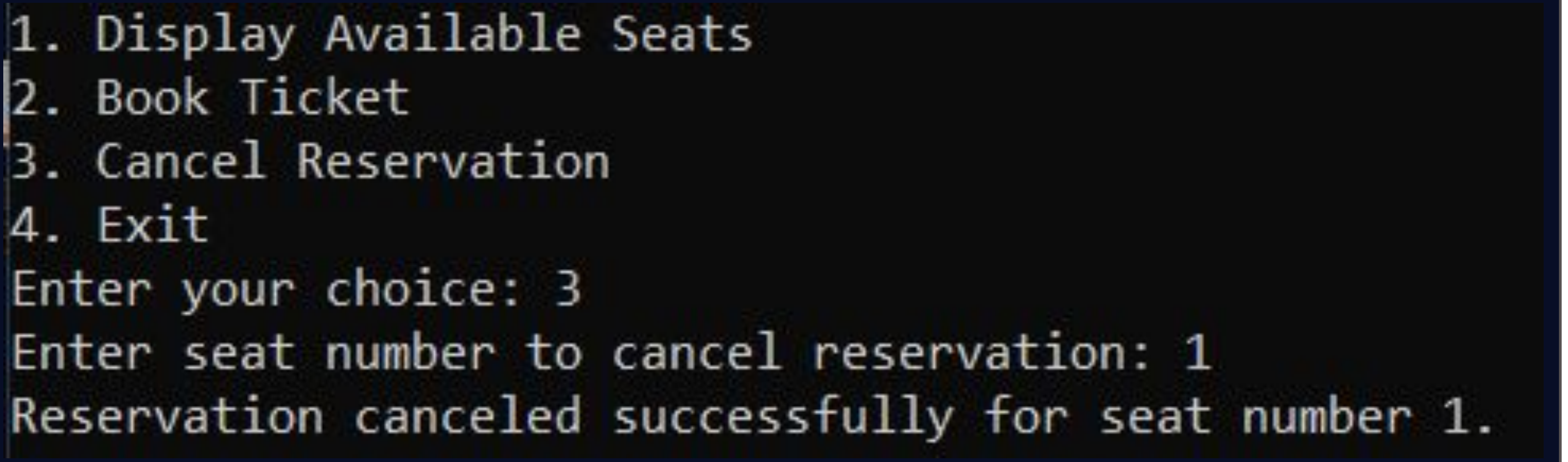
```
1. Display Available Seats
2. Book Ticket
3. Cancel Reservation
4. Exit
Enter your choice: 1
Available Seats: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 3
8 39 40 41 42 43 44 45 46 47 48 49 50
```

iv) Booking tickets for seat



```
1. Display Available Seats
2. Book Ticket
3. Cancel Reservation
4. Exit
Enter your choice: 2
Enter seat number to book: 1
Ticket booked successfully for seat number 1.
```

v) Cancel the Reservation



```
1. Display Available Seats
2. Book Ticket
3. Cancel Reservation
4. Exit
Enter your choice: 3
Enter seat number to cancel reservation: 1
Reservation canceled successfully for seat number 1.
```

Conclusion and Contributions:

With our Railway Reservation System, we aim to revolutionize the way people book and manage train tickets.

Contributions:

Mrunal Anandanche:

Implementation of structures and main(), displaySchedule()

Nikita Anup:

Implemented createSchedule(), createReservation() and displayAvailableSeats()

Nikitha T:

Implemented bookTicket(), cancelReservation() and freeReservations()