K.RAMAKRISHNAN COLLEGE OF ENGINEERING
(AUTONOMOUS)

ONLINE TRAIN TICKET BOOKING SYSTEM

PRESENTED BY
NAME : S.SAKTHI
REGISTER NO: 8115U23AD050
ROLL NO: AD23050
YR/SEC: I/E

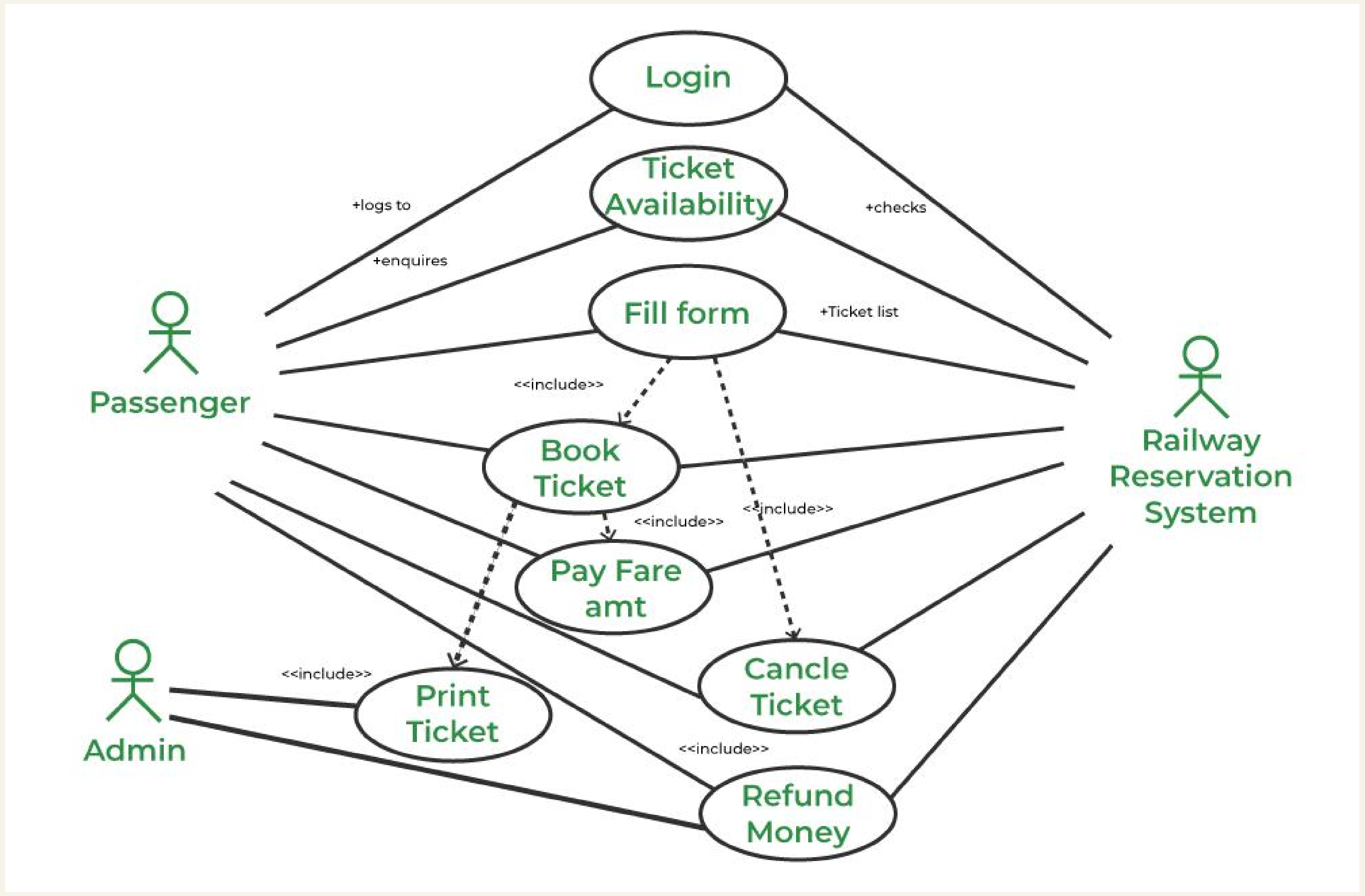
PRESENTATION OVERVIEW

- OBJECTIVE
- SYSYTEM ARCHITECTURE(flow diagram)
- DATA STRUCTURE USED
- MODULE COMPLETION
- CONCLUSION

OBJECTIVE

The objective of an online train ticket booking system project is to develop a comprehensive, user-friendly web-based application that facilitates the booking and management of train tickets. This system aims to simplify the process for passengers, offering them a convenient way to search for train schedules, check seat availability, and make secure bookings online. Additionally, the system provides administrative tools for managing train schedules, routes, and bookings.

ARCHITECTURE DIAGRAM

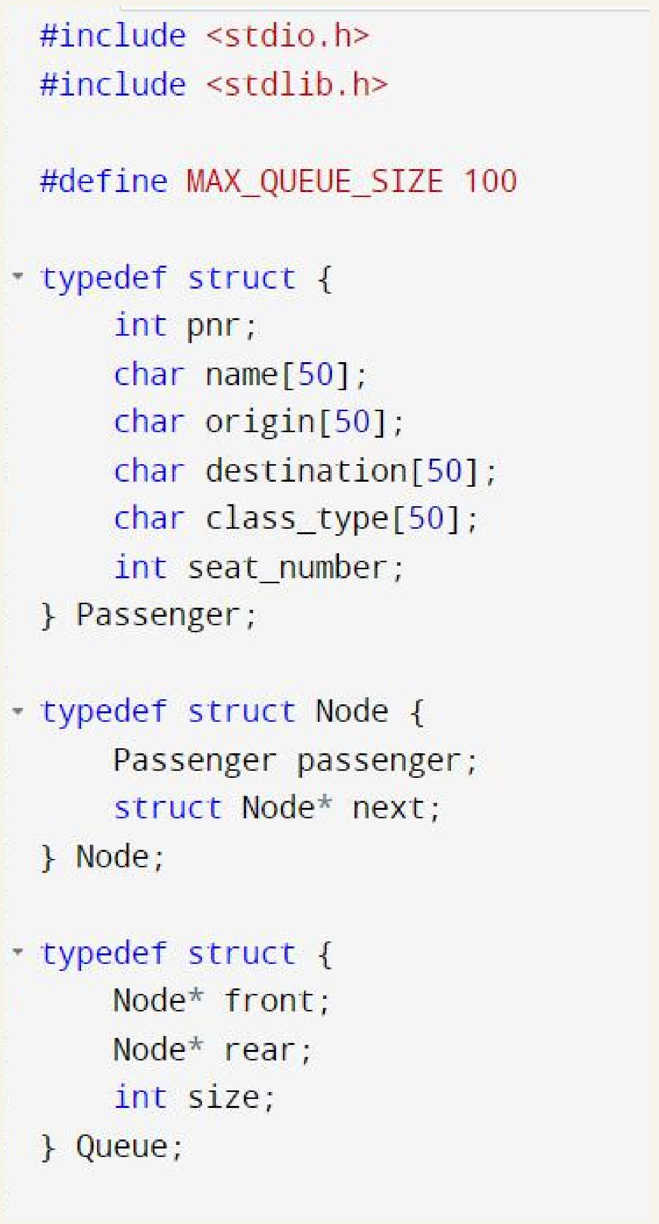


DATA STRUCTURE USED

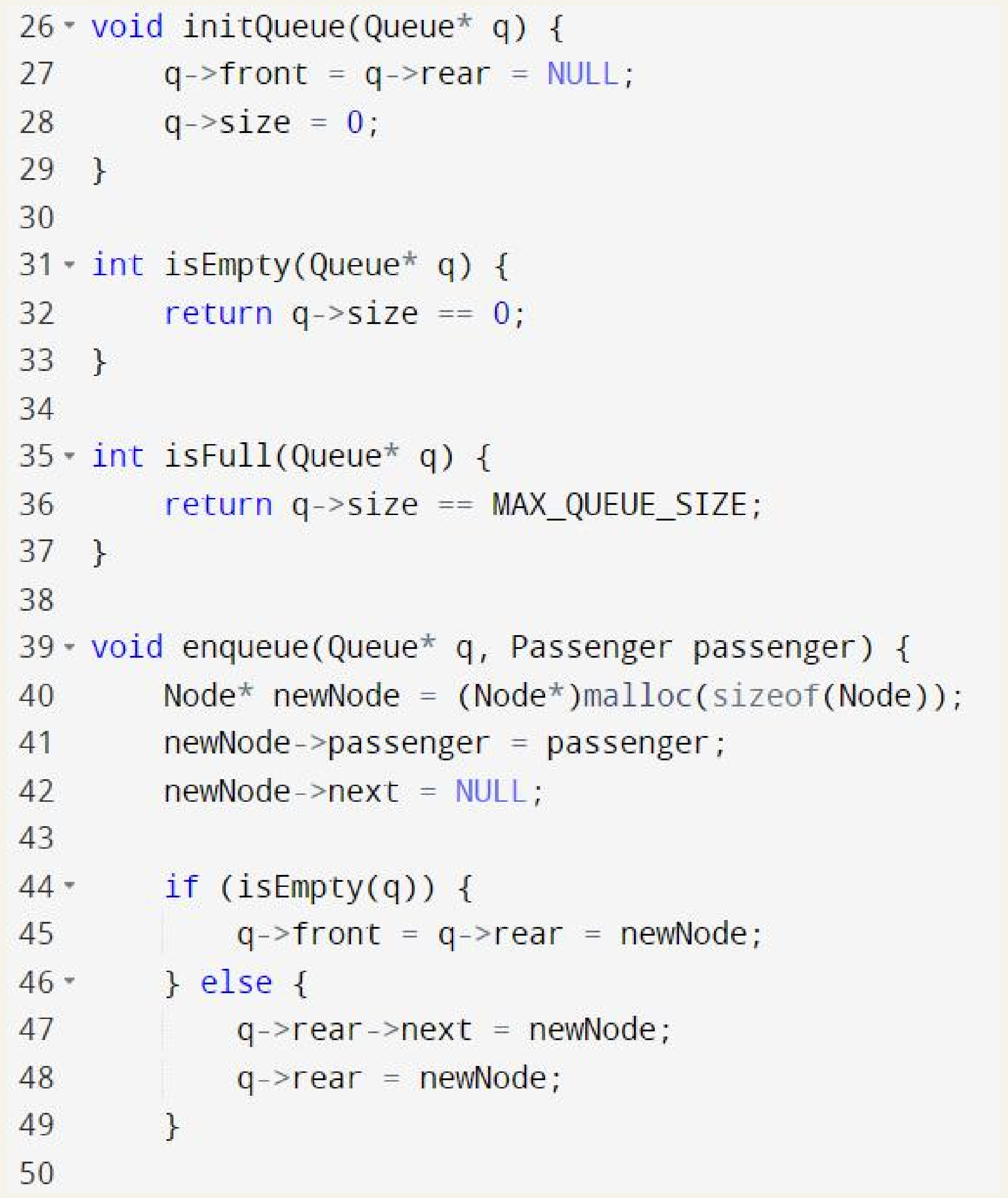
The array implementation of a queue is a common method used to represent a queue data structure. In this implementation, a fixed-size array is used to store the elements of the queue.

When implementing a queue using an array, two indices are used: one for the front of the queue and another for the rear. The front index points to the first element in the queue, while the rear index points to the last element.
When an element is enqueued (added) to the queue, it is inserted at the rear index, and the rear index is incremented. When an element is dequeued (removed) from the queue, it is removed from the front index, and the front index is incremented to the next element in the queue.

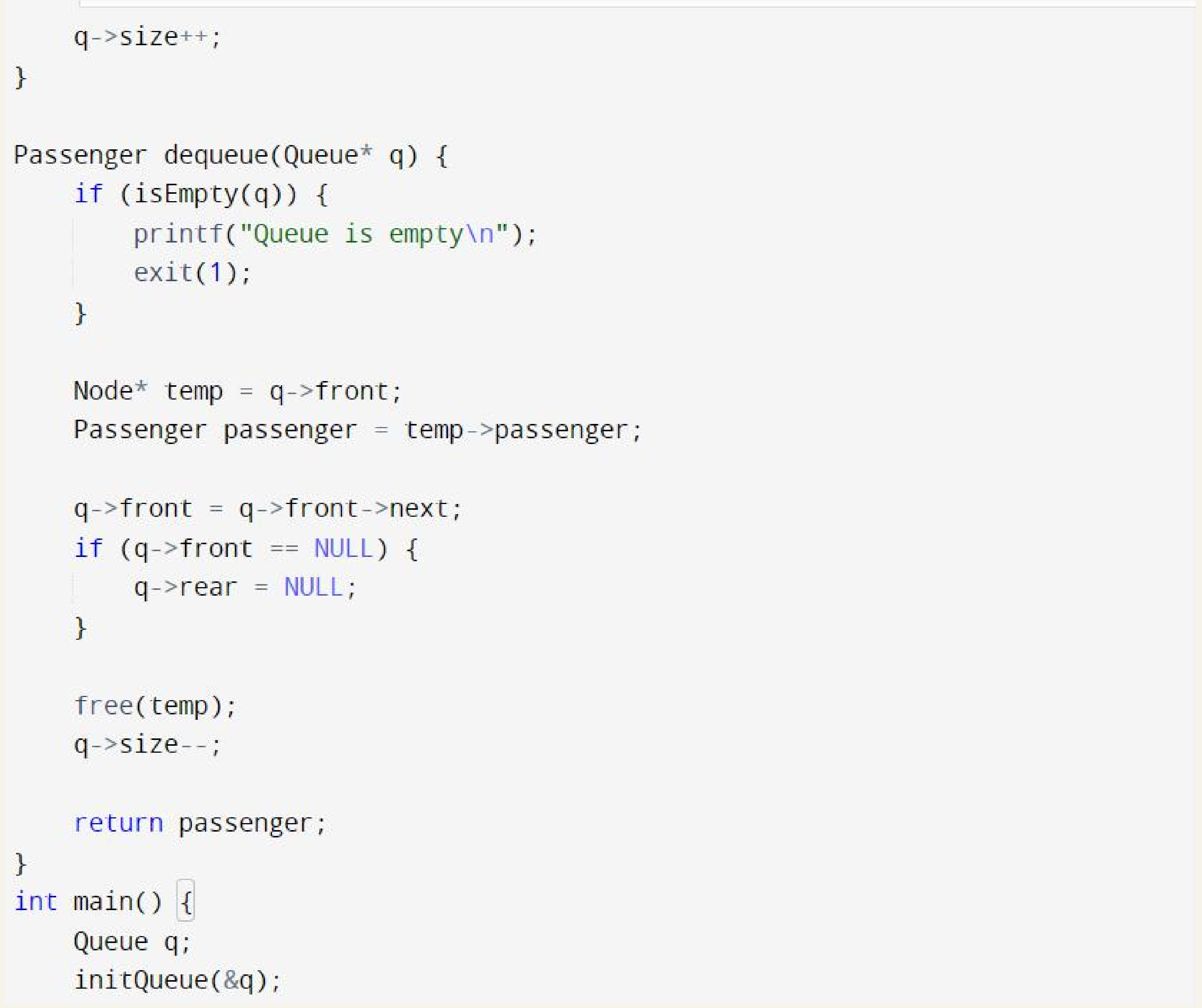
MODULE COMPLETION



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

#define MAX_QUEUE_SIZE 100

typedef struct {
    int pnr;
    char name[50];
    char origin[50];
    char destination[50];
    char class_type[50];
    int seat_number;
} Passenger;

typedef struct Node {
    Passenger passenger;
    struct Node* next;
} Node;

typedef struct {
    Node* front;
    Node* rear;
    int size;
} Queue;
```

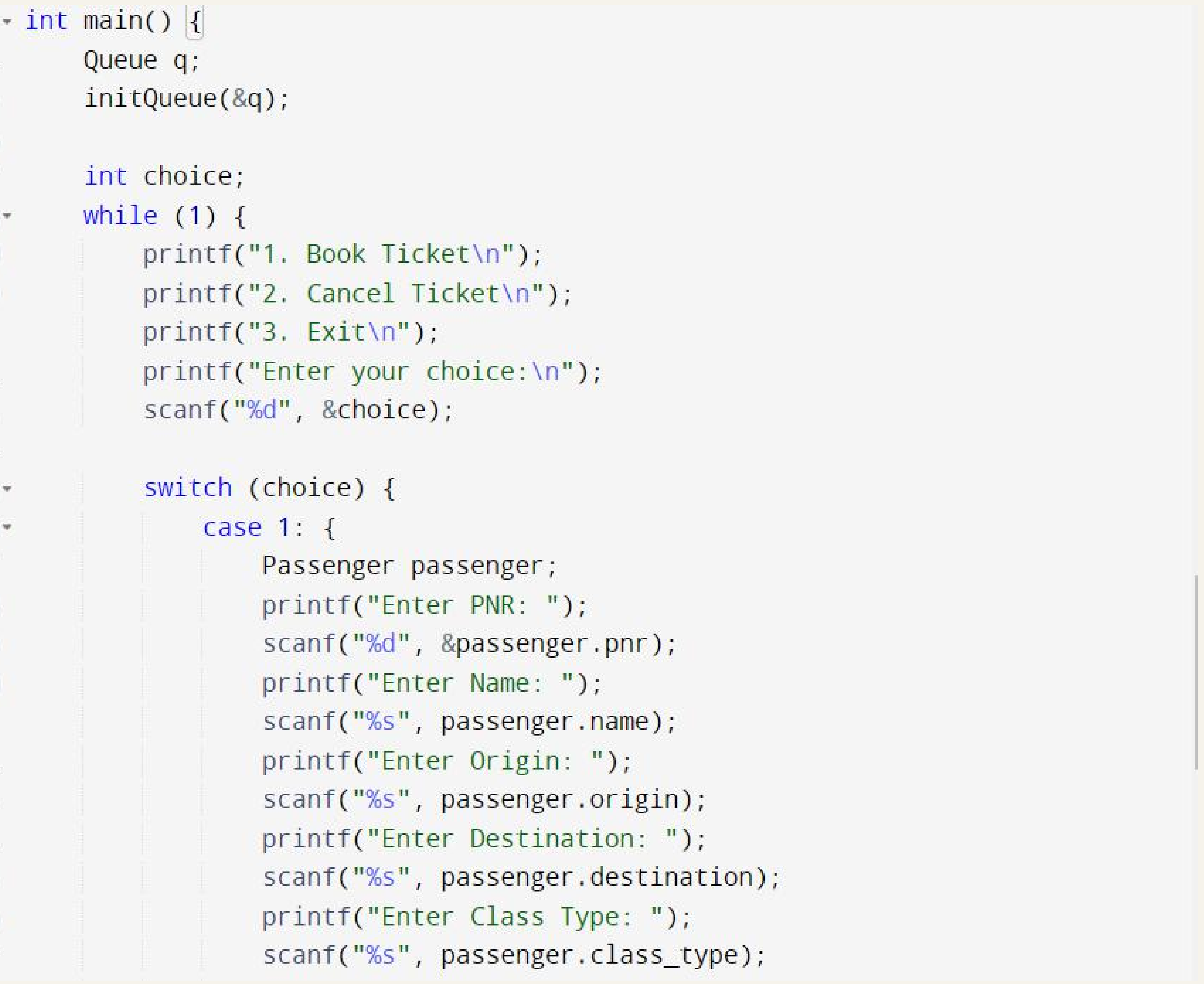


```c
void initQueue(Queue* q) {
    q->front = q->rear = NULL;
    q->size = 0;
}

int isEmpty(Queue* q) {
    return q->size == 0;
}

int isFull(Queue* q) {
    return q->size == MAX_QUEUE_SIZE;
}

void enqueue(Queue* q, Passenger passenger) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->passenger = passenger;
    newNode->next = NULL;

    if (isEmpty(q)) {
        q->front = q->rear = newNode;
    } else {
        q->rear->next = newNode;
        q->rear = newNode;
    }

```

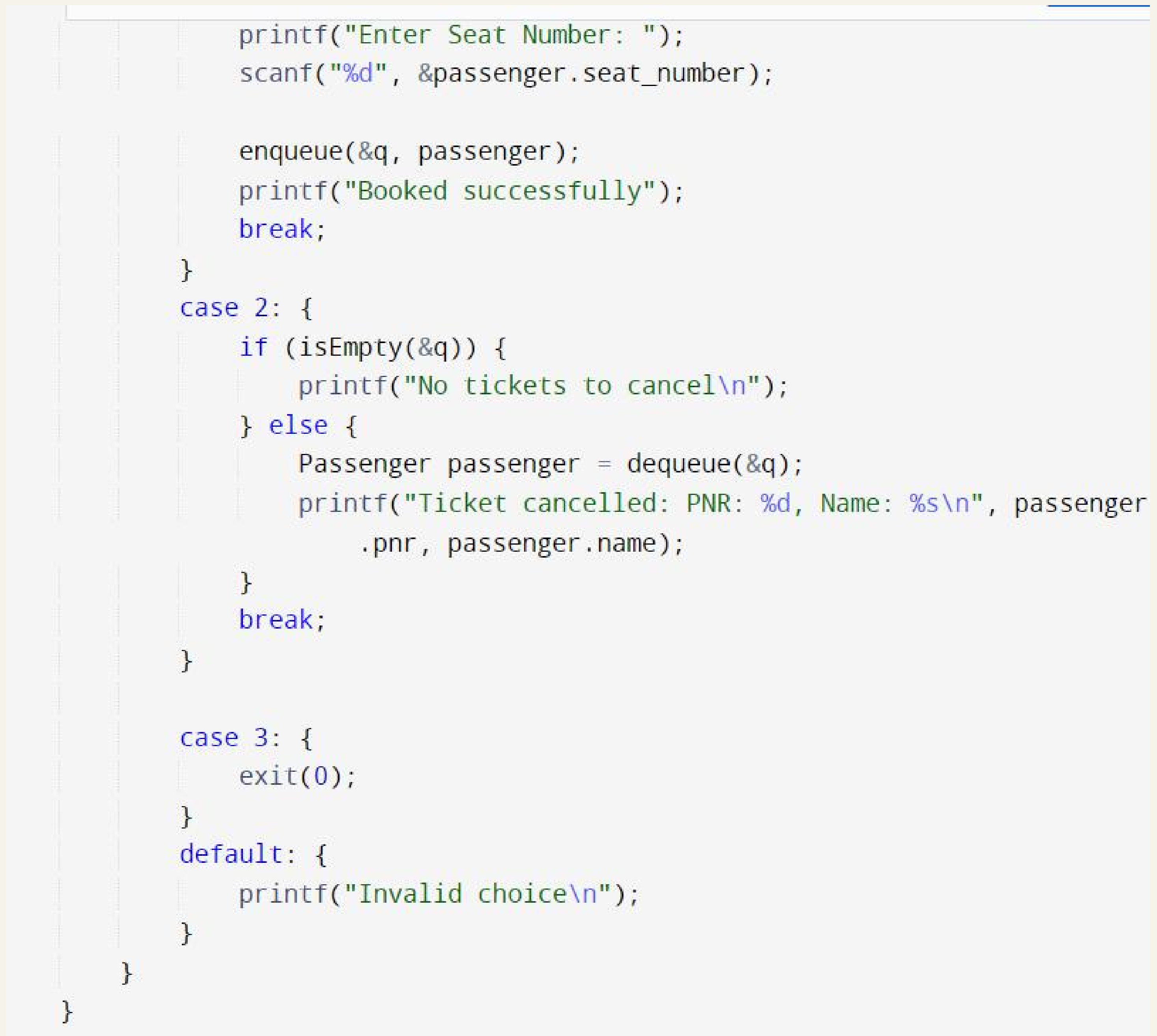


```c
    q->size++;
}

Passenger dequeue(Queue* q) {
    if (isEmpty(q)) {
        printf("Queue is empty\n");
        exit(1);
    }

    Node* temp = q->front;
    Passenger passenger = temp->passenger;

    q->front = q->front->next;
    if (q->front == NULL) {
        q->rear = NULL;
    }

    free(temp);
    q->size--;

    return passenger;
}
int main() {
    Queue q;
    initQueue(&q);
```

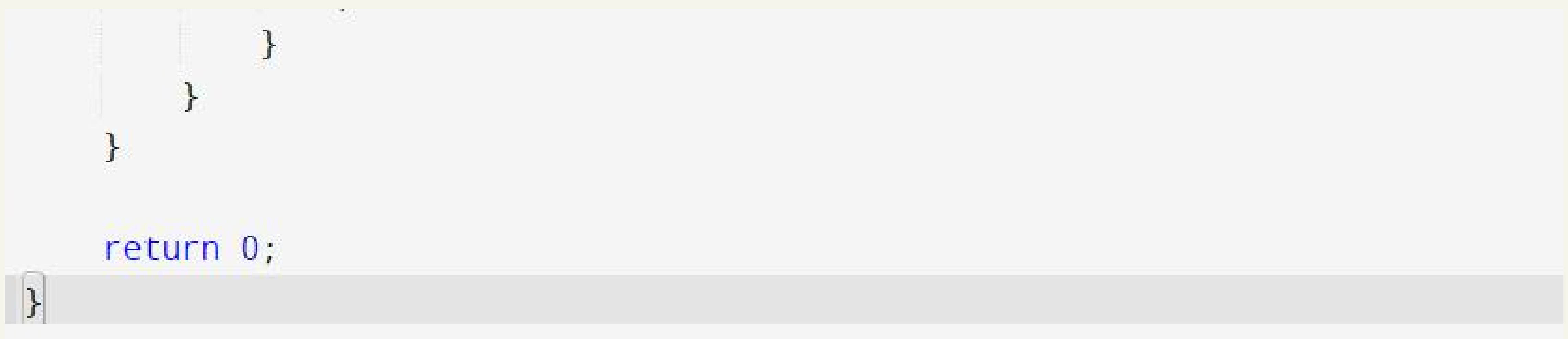
MODULES COMPLETED



```c
int main() {
    Queue q;
    initQueue(&q);

    int choice;
    while (1) {
        printf("1. Book Ticket\n");
        printf("2. Cancel Ticket\n");
        printf("3. Exit\n");
        printf("Enter your choice:\n");
        scanf("%d", &choice);

        switch (choice) {
            case 1: {
                Passenger passenger;
                printf("Enter PNR: ");
                scanf("%d", &passenger.pnr);
                printf("Enter Name: ");
                scanf("%s", passenger.name);
                printf("Enter Origin: ");
                scanf("%s", passenger.origin);
                printf("Enter Destination: ");
                scanf("%s", passenger.destination);
                printf("Enter Class Type: ");
                scanf("%s", passenger.class_type);
```

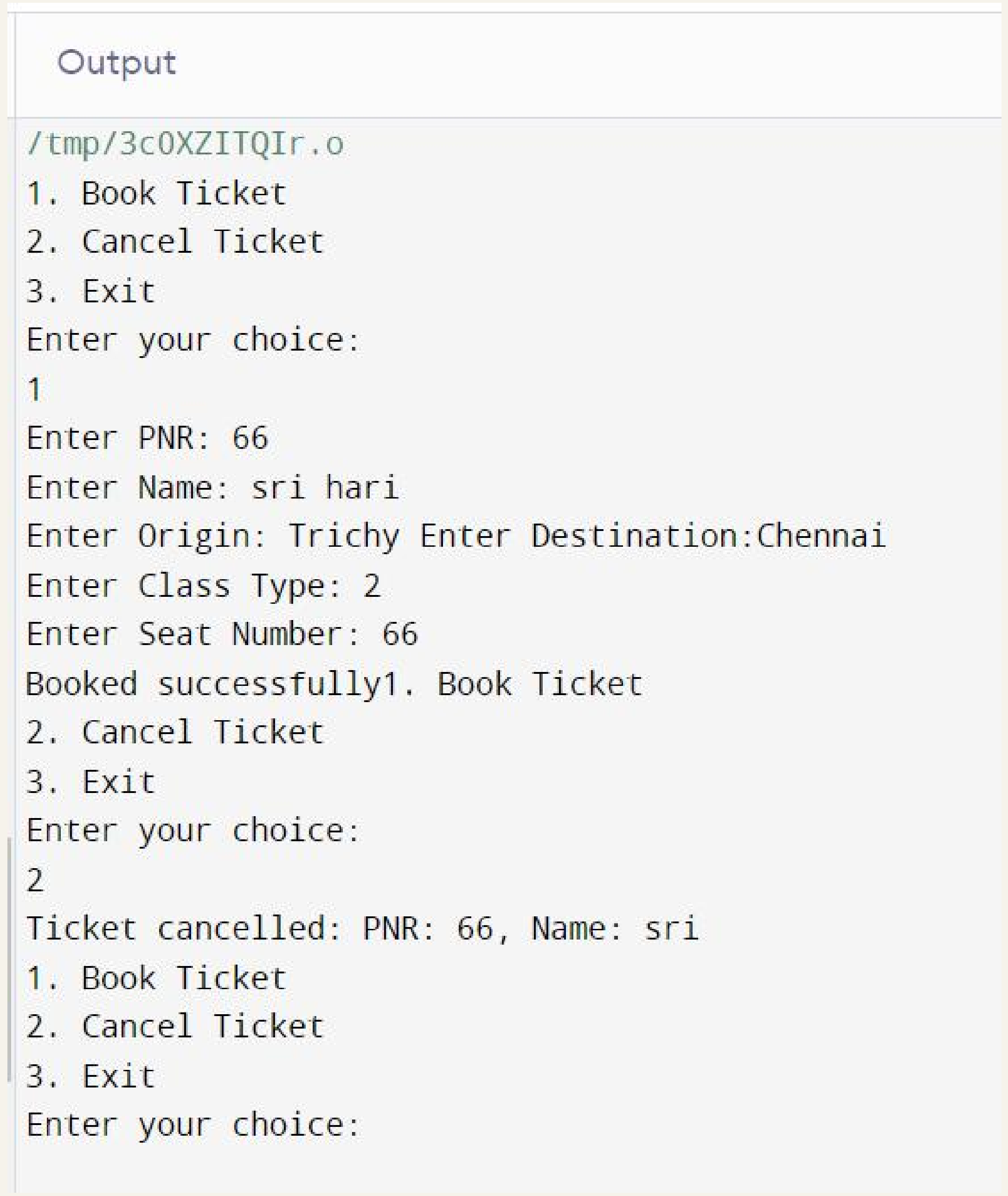


```c
                printf("Enter Seat Number: ");
                scanf("%d", &passenger.seat_number);

                enqueue(&q, passenger);
                printf("Booked successfully");
                break;
            }
            case 2: {
                if (isEmpty(&q)) {
                    printf("No tickets to cancel\n");
                } else {
                    Passenger passenger = dequeue(&q);
                    printf("Ticket cancelled: PNR: %d, Name: %s\n", passenger
                        .pnr, passenger.name);
                }
                break;
            }

            case 3: {
                exit(0);
            }
            default: {
                printf("Invalid choice\n");
            }
        }
    }
```



```c
            }
        }
    }

    return 0;
}
```

OUTPUT



```
/tmp/3c0XZITQIr.o
1. Book Ticket
2. Cancel Ticket
3. Exit
Enter your choice:
1
Enter PNR: 66
Enter Name: sri hari
Enter Origin: Trichy Enter Destination:Chennai
Enter Class Type: 2
Enter Seat Number: 66
Booked successfully1. Book Ticket
2. Cancel Ticket
3. Exit
Enter your choice:
2
Ticket cancelled: PNR: 66, Name: sri
1. Book Ticket
2. Cancel Ticket
3. Exit
Enter your choice:
```

MODULE EXPLANATION

typedef struct { ... } Passenger;: Defines a structure to store passenger details.

typedef struct Node { ... } Node;: Defines a node structure for the linked list.

typedef struct { ... } Queue;: Defines a queue structure that uses the linked list.

void initQueue(Queue* q);: Initializes the queue.

int isEmpty(Queue* q); : Checks if the queue is empty.

int isFull(Queue* q);: Checks if the queue is full.

void enqueue(Queue* q, Passenger passenger);: Adds a passenger to the queue.

Passenger dequeue(Queue* q);: Removes a passenger from the queue.

THANK YOU