PIMPRI CHINCHWAD EDUCATION TRUST's.

PIMPRI CHINCHWAD COLLEGE OF ENGINEERING

(An Autonomous Institute)

Class : SY BTech **Acad. Yr. 2025-26** **Semester : I**

Name of the student: Om Jitendra Khalane **PRN :** 124B1B040

Department: Computer Engineering **Division :** A

Course Name : Data Structures and Laboratory

Course Code: BCE23PC02

Completion Date : 12/08/2025

Problem Statement:

Design a **simplified Railway Reservation System** using **arrays** and **queues**.
The system should allow users to:

1. **Book tickets** (confirmed if seats are available, otherwise added to a waiting list).
2. **Cancel tickets**, which frees up a seat and moves the first passenger from the waiting list to confirmed booking.
3. **View all tickets**, showing both confirmed bookings and the waiting list.

Use an **array** for confirmed tickets and a **queue** (linked list or circular linked list) for the waiting list.

Source Code :

https://github.com/omkhalane/DSAL-SY-PCCOE/blob/main/lab_assignments/assignment09.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
```

```
class Queue
{
    int ID;
    string name;
    Queue *next = nullptr;

public:
    Queue() {}
    Queue(string n, int id)
    {
        name = n;
        ID = id;
    }
    friend class Ticket_system;
};

class Ticket_system
{
    static const int MAX = 5;
    string booked[MAX];
    int bookedCount = 0;

    int identy = 1;
    Queue *front = nullptr, *rear = nullptr;

public:
    void add_ticket(string name)
    {
        if (bookedCount < MAX)
        {
            booked[bookedCount++] = name;
```

```
                cout << "Ticket confirmed for " << name << " (ID " << identy++
<< ")\n";
            }
            else
            {
                Queue *temp = new Queue(name, identy++);
                if (rear == nullptr)
                {
                    front = rear = temp;
                }
                else
                {
                    rear->next = temp;
                    rear = temp;
                }
                cout << "All seats full! Added to waiting list: " << name <<
endl;
            }
        }

        void remove_ticket()
        {
            if (bookedCount == 0)
            {
                cout << "No confirmed bookings to cancel!\n";
                return;
            }

            cout << "Cancelled ticket of " << booked[0] << endl;

            for (int i = 1; i < bookedCount; i++)
                booked[i - 1] = booked[i];
```

```
            bookedCount--;

            if (front != nullptr)
            {
                booked[bookedCount++] = front->name;
                cout << "Moved from waiting list to confirmed: " << front->name
<< endl;

                Queue *temp = front;
                front = front->next;
                delete temp;
                if (front == nullptr)
                    rear = nullptr;
            }
        }

        void print_all()
        {
            cout << "\n--- Confirmed Tickets ---\n";
            if (bookedCount == 0)
                cout << "None\n";
            else
                for (int i = 0; i < bookedCount; i++)
                    cout << i + 1 << ". " << booked[i] << endl;

            cout << "\n--- Waiting List ---\n";
            if (front == nullptr)
                cout << "None\n";
            else
            {
                Queue *temp = front;
                while (temp != nullptr)
```

```
            {
                cout << temp->ID << ". " << temp->name << endl;
                temp = temp->next;
            }
        }
        cout << endl;
    }
};

int main()
{
    Ticket_system obj;
    cout << "=== Railway Reservation System ===\n";
    cout << "1. Book Ticket\n";
    cout << "2. Cancel Ticket\n";
    cout << "3. View All Tickets\n";
    cout << "4. Exit\n";

    while (true)
    {
        int op;
        cout << "\nEnter Option: ";
        cin >> op;
        cin.ignore();

        switch (op)
        {
        case 1:
        {
            string name;
            cout << "Enter Passenger Name: ";
            getline(cin, name);
```

```
                obj.add_ticket(name);
                break;
            }
            case 2:
                obj.remove_ticket();
                break;
            case 3:
                obj.print_all();
                break;
            case 4:
                return 0;
            default:
                cout << "Invalid option!\n";
                break;
            }
        }
    }
```

Conclusion:

- **Confirmed Tickets:** Stored in a fixed-size array for easy indexing and management.
- **Waiting List:** Managed using a **queue**, implemented with a linked list, to maintain first-come-first-serve order.
- Booking moves a passenger to **confirmed array** if seats are available; otherwise, the passenger is added to the **waiting queue**.
- Cancellation removes the passenger from the confirmed array and automatically moves the next waiting passenger to confirmed.

Time Complexity (TC):

- **Booking (Array insertion):** O(1)
- **Cancellation (Shift elements in array):** O(n)
- **Waiting List Enqueue/Dequeue:** O(1)
- **View Tickets:** O(n + m) — n confirmed tickets, m waiting passengers

Space Complexity (SC):

- **O(MAX + m)** — MAX seats for confirmed tickets in the array, m nodes in waiting list queue.