Data Structures

Assignments

Instructions:

- Solve all the given assignments below.
- **Assignments are for all regular as well as backlog students.**
- Write C code with proper input and output
- Submit your assignment in .PDF format only. PDF File consists: Problem Statement, C-Code, and Input/output.
- Last date of Submission 24th April 2025 till 5PM.
- Send your assignment on: sksahu.exam@gmail.com for evaluation.
- Late submission will not be considered.

Problem Statement 1

Problem Statement: Railway Reservation System

Real-World Scenario:

Indian Railways manages thousands of passengers daily. A common task is to handle seat reservations and cancellations efficiently. Design a railway reservation system that:

1. Maintains a waitlist for passengers when seats are unavailable.
2. Assigns seats to passengers from the waitlist when cancellations occur.
3. Displays the current status of reserved seats and the waitlist.

Relevance to Syllabus:

- **Arrays:** Store reserved seat numbers.
- **Queues:** Implement the waitlist using a queue (linked list-based).
- **Linked Lists:** Manage dynamic addition/removal of passengers in the waitlist.
- **Basic Operations:** Insertion, deletion, and traversal align with Modules 1, 2, and 3.

Problem Details:

- The train has a fixed number of seats (e.g., 5 for simplicity).
- If all seats are reserved, new passengers are added to a waitlist.
- If a passenger cancels their reservation, the first person from the waitlist gets the seat.
- Operations: Reserve a seat, Cancel a reservation, and Display status.

Example:

Seat reserved for Amit (PNR: 1001, Seat: 1).
Seat reserved for Priya (PNR: 1002, Seat: 2).
Seat reserved for Rohan (PNR: 1003, Seat: 3).
...
Seat reserved for Rahul (PNR: 1009, Seat: 9).
Passenger Anjali (PNR: 1010) added to waitlist.
Passenger Suresh (PNR: 1011) added to waitlist.

Reserved Seats:
Seat 1: Amit (PNR: 1001)
Seat 2: Priya (PNR: 1002)
...
Seat 9: Rahul (PNR: 1009)

Waitlist:
Anjali (PNR: 1010)
Suresh (PNR: 1011)

Cancelled reservation for Amit (PNR: 1001, Seat: 1).
Seat assigned to Anjali (PNR: 1010, Seat: 1) from waitlist.

Reserved Seats:
Seat 1: Anjali (PNR: 1010)
Seat 2: Priya (PNR: 1002)
...

Waitlist:
Suresh (PNR: 1011)

Problem Statement 2

Problem Statement: Undo and Redo in a Text Editor

Real-World Scenario:

A text editor allows users to type lines of text and provides "Undo" and "Redo" functionality. When a user adds a line, they can undo it to remove the last addition, and redo it to reapply the undone action. Design a system to:

1. Add a new line of text to the editor.
2. Undo the last added line (move it to a redo stack).
3. Redo the last undone line (move it back to the editor).
4. Display the current text in the editor.

Syllabus Alignment:

- **Module 3:** Stacks (ADT, Push, Pop, Implementation using Linked Lists).
- **Real-World Relevance:** Simulates features in text editors like Notepad++, Microsoft Word, or Google Docs, where undo/redo is essential for user experience.

Requirements:

- Use two stacks: one for the current text (undo stack) and one for undone actions (redo stack).
- Support dynamic memory allocation (Module 2) for scalability.

• Problem Statement 3

•

- **Find at least 5 best real time problem statement** related to data structures (Try to select problem statements from each module wise) from standard coding platforms (Leetcode, Codechef, HackerRank, Codeforces) and provide optimal solution with proper input and output.