Data Structures

3 Part – Team Lab

This lab provides information for Lab 1.

Lab 1 — Lines & Logs (Stacks, Queues, Linked Lists)

Scenario: You’re building the core engine for an “Operations Control” simulator for a busy airport terminal. Events arrive (arrivals, gate changes, cleaning requests, boarding calls), and your system must process them in order, support undo, and keep an editable, ordered roster.

Learning goals

* Implement and compare Queue, Stack, and Singly Linked List.
* Practice enqueue/dequeue, push/pop, insert/delete at index, traversal.
* Separate interfaces from implementations; write small black-box tests.

Team roles (you might want to rotate these for the 3 parts)

* Lead dev (coordinates interfaces)
* Data structure owner A (Queue/Stack)
* Data structure owner B (Linked List)
* Tester/Tech writer (CLI + tests + README)

Required features

1. EventQueue (FIFO)
   * enqueue(event), dequeue(), peek(), is\_empty(), \_\_len\_\_().
   * Backed by your own list/array or linked list nodes (no library queue).
2. UndoStack (LIFO)
   * push(cmd), pop(), peek(), is\_empty().
   * Stores command objects ( “insert passenger at index 3”).
3. Roster (Singly Linked List)
   * append(name), insert(index, name), remove(index), find(name), iter()
   * Head/tail pointers; O(1) tail append if you keep tail.

Dataset (starter) Use persistent storage with JSON file (research)

sample\_events.json

[

{"type":"arrive","name":"Ana"},

{"type":"arrive","name":"Ben"},

{"type":"insert","name":"Cam","index":1},

{"type":"remove","index":0},

{"type":"arrive","name":"Dia"}

]

Deliverables

* structures/queue.py, structures/stack.py, structures/linkedlist.py
* sim/core.py (event loop), tests/test\_lab1.py, README.md

Rubric (100 pts)

* Queue correct & tested (25)
* Stack correct & tested (25)
* Linked list ops correct (25)
* CLI + README + style (25)

Adv Option (for more efficient solution): doubly linked list for O(1) delete with node handle; batch UNDO.