**Lab 4 Discrete**

**Member 1:** Ibrahim Mohamed Elsayed Moustafa Omar|21010023

**Member 2:** Mohamed Mohamed Mohamed Abdelmonaem|21011213

**Problem 1 (Airline Network Shortest-Path Finder):**

**I] Problem Statement:**

Imagine you are tasked with developing a tool to assist airline passengers in finding the most efficient route between two airports within the airline's network. Create a Java program that models an airline network as a graph, where airports are nodes and flights are edges. Your program should enable users to input details of the flight connections between airports and find the shortest path between a specified source and destination airport.

• Implement a class to represent the airline network graph. Each airport is a

node, and flights between airports are edges. You can choose an adjacency

matrix or adjacency list to represent the graph.

• Implement Dijkstra's algorithm to find the shortest path between two specified

airports in the airline network.

• Display the optimal route details, including the sequence of airports to visit

and the total distance or time required for the journey.

• Implement error handling mechanisms to handle cases where the specified

source or destination airport is not in the network or when there is no direct

flight between them.

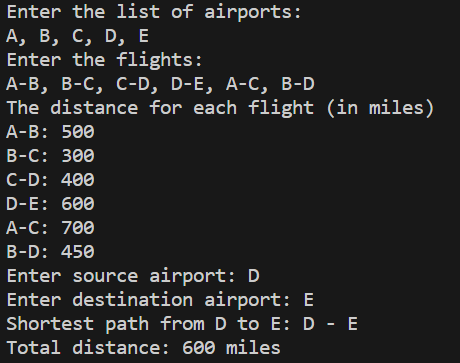
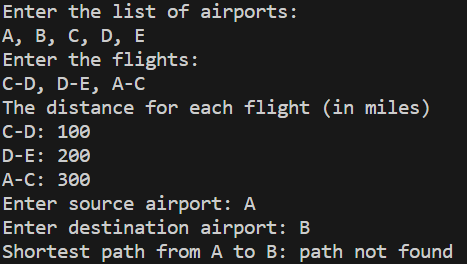
**II] Data Structures:**

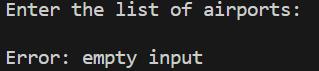
* Array List

**III] Sample runs/test cases:**

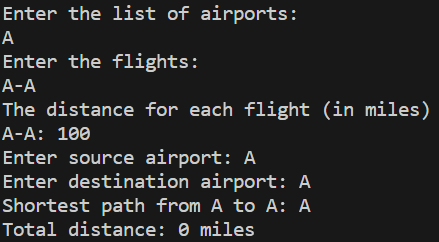
A screen shot of a computer program

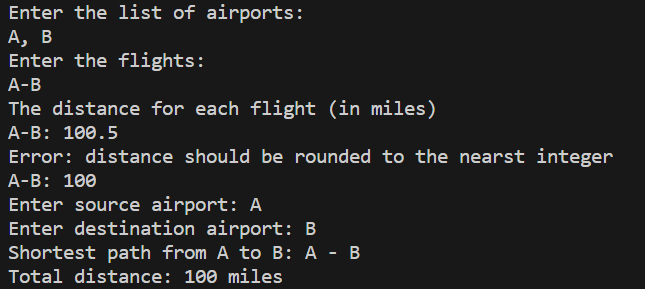
Description automatically generated

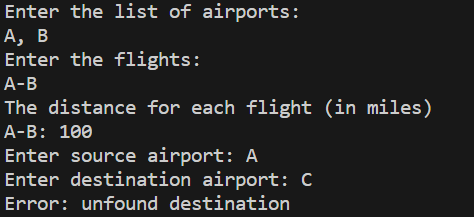


A black screen with white text

Description automatically generated







**IV] Assumptions:**

* Distances between airports must be rounded to the nearest integer.

**Problem 2 (Class Schedule Optimization):**

**I] Problem Statement:**

**II] Data Structures:**

**III] Sample runs/test cases:**

**IV] Assumptions:**

**Problem 3 (Tree Traversal):**

* **Problem Statement:**

Implement three algorithms for Binary Tree traversal recursively or iteratively:

• Preorder

• Inorder

• Postorder

**I] Data Structures:**

* Linked Transfer Queue. 🡪 in reading tree

**II] Sample runs/test cases:**

**III] Assumptions:**