**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.

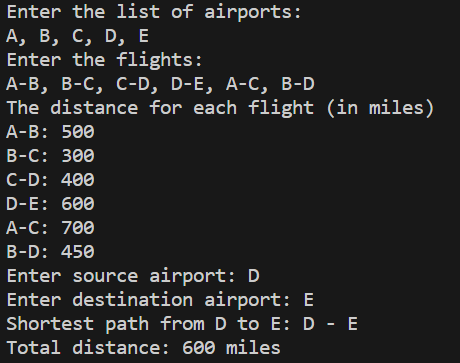
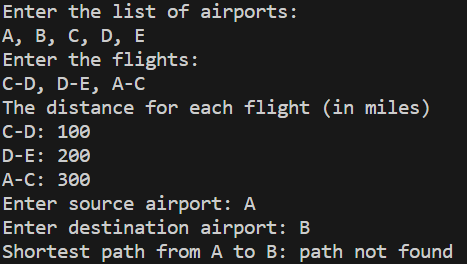
**II] Data Structures:**

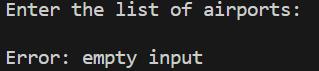
* Array List

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

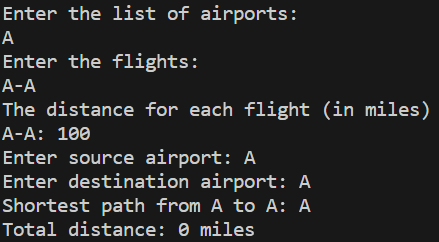
A screen shot of a computer program

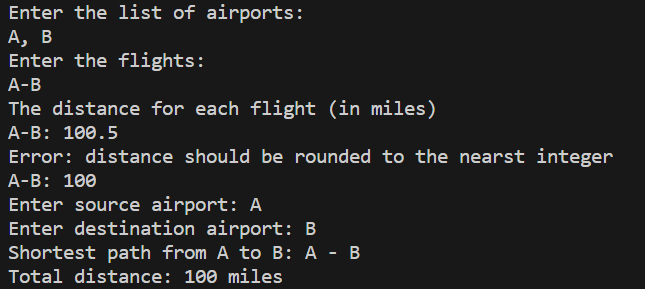
Description automatically generated

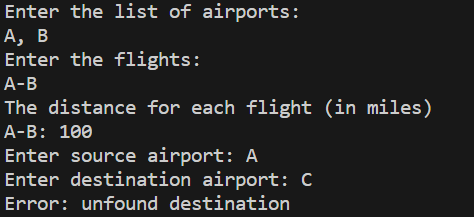


A black screen with white text

Description automatically generated







**IV] Assumptions:**

* Distances between airports must be rounded to the nearest integer.
* If repeated nodes (classes) are found by false input, repetition will be removed.

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

**I] Problem Statement:**

Imagine a school with multiple classes and subject timings where certain classes cannot occur simultaneously due to shared resources or teacher availability. Develop a Java program that generates an optimized class schedule by assigning time slots to classes, ensuring that no conflicting classes occur at the same time.

• Represent the schedule information as a graph where nodes represent classes, and edges denote conflicting timings between classes.

• Implement a graph coloring algorithm to assign distinct colors (time slots) to nodes (classes) in the graph. Ensure that adjacent nodes (classes) linked by edges (conflicting timings) do not share the same color (time slot) to avoid scheduling conflicts.

• Display the timetable with color-coded class timings, ensuring that conflicting classes have different colors (non-overlapping timings).

• Use any color names as you like.

**II] Data Structures:**

* Array List

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

**A screen shot of a computer

Description automatically generatedA screen shot of a computer program

Description automatically generated**

**A screen shot of a computer program

Description automatically generated** **A screen shot of a computer

Description automatically generated**

**IV] Assumptions:**

* Conflicting classes must be enters on one line separated by “,”
* If repeated nodes (classes) are found by false input, repetition will be removed.
* If conflicting classes are left empty (enter click) that means that there are no conflicting classes.

**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:**

**A screenshot of a computer program

Description automatically generated**

**A screen shot of a computer

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**A black background with white text

Description automatically generated**

**A black background with white text

Description automatically generatedA black background with white text

Description automatically generatedA black background with white text

Description automatically generated**

**III] Assumptions:**

* Input (node value) must be positive integer, except for -1 to skip node.