DSA Hackathon Problem Statement

Problem Statement: Social Network Friend Recommendation System

Create a program to manage a social network of users, where users can establish and query connections, find new friends based on shared contacts, and access various functionalities.

Input Format:

1. **Initial Setup**: The network is initialized with users and their connections based on input data.
    - 7 // Number of users
    - Alice Bob Charlie Dave Eve Karen John // List of user names
    - 7 // Number of friendships
    - Alice Bob
    - Alice Eve
    - Alice Dave
    - Alice Charlie
    - Bob Charlie
    - Bob John
    - Karen Dave

Functionalities:

Display Network Structure:

- **Input**: print_network
- **Output**: Display all users and their connections.
  Example:
  Alice: Bob, Eve, Dave, Charlie
  Bob: Alice, Charlie, John
  Charlie: Bob, Alice
  Dave: Alice, Karen
  Eve: Alice
  Karen: Dave
  John: Bob

List Friends of a User:

- **Input**: list_friends <user>
- **Output**: List all friends of the specified <user>.
  Example:
  Friends of Charlie:
  - Alice
  - Bob

Find Mutual Friends:

- **Input**: mutual_friends <user1> <user2>
- **Output**: Display mutual connections between <user1> and <user2>.
  Example:
  Mutual friends between Alice and Bob:
  - Charlie

Find Possible Friends (Recommendations):

- **Input**: recommend_friends <user> <degree>
- **Output**: Suggest friends for <user> by finding users connected within <degree> levels (e.g., friends of friends).
  Example:
  Friend recommendations for Alice within 2 levels:
  - Karen
  - John

Identify Popular Users:

- **Input**: most_popular
- **Output**: List users based on the number of their connections, in descending order.
  Example:
  Most popular users:
  - Alice (4 connections)

Find Shortest Path Between Users:

- **Input**: shortest_path <user1> <user2>
- **Output**: Display the shortest path from <user1> to <user2> if it exists, showing the sequence of connections.
  Example:
  Shortest path from John to Dave:
  John -> Bob -> Alice -> Dave

Constraints:

- The network's structure should be provided as an initial input.
- The program should handle cases where no connection exists between users.
- The solution should manage and query connections efficiently.