CASH FLOW OPTIMIZATION

Working of the Project

The project is designed to minimize the number of financial transactions needed among multiple entities, either banks or individuals, to settle their debts. The primary goal is to reduce the complexity and number of transactions by optimizing how these debts are settled, considering constraints such as different payment modes for banks and all available payment methods for personal transactions.

1. Main Branch (main() Function)

1.1 Input Initialization

- **Purpose:** Gather input details about the participants, including the number of banks or individuals, their payment modes (for banks), and the transactions between them.
- **Key Operations:**
  - For personal transactions, gather the number of individuals, their names, and the transaction details.
  - For bank transactions, gather information about each bank, the payment modes they support, and the transaction details.
- **Flow:**
  - Calls `minimizeCashFlow_personal()` for personal transactions or `minimizeCashFlow_Bank()` for bank transactions based on user selection.

1.2 Data Collection

- **Purpose:** Collect and store data for the transaction processing.
- **Structure:**
  - For personal transactions: A list of individuals and their net balances.
  - For bank transactions: An adjacency matrix representing the amount owed between banks.

2. Core Calculation Branch

2.1 Net Amount Calculation (for Banks)

- **Purpose:** Calculate the net balance for each bank, which represents the difference between the total incoming and outgoing amounts.
- **Flow:**
  - Iterate through each bank to compute net balances based on the adjacency matrix.

2.2 Transaction Minimization Loop (for Banks)

- **Purpose:** Minimize the number of transactions required to settle all debts among banks.
- **Key Steps:**
  - **Find Bank with Minimum Net Amount:** Identify the bank with the smallest negative balance.

    - **Find Optimal Creditor:** Identify a creditor bank with a positive balance that shares at least one payment mode with the debtor.
    - **Perform Transaction:** Adjust balances based on the transaction amount and record transactions in an answer matrix.
- **Flow:** Continues until all banks have a net amount of zero.

2.3 **Output the Minimized Transactions (printAns())**

- **Purpose:** Display the optimized list of transactions that minimize the number of exchanges required.
- **Flow:** Traverse the answer matrix and print the transactions.

3. Utility Branches (Helper Functions)

3.1 **getMinIndex_bank()**

- **Purpose:** Find the index of the bank with the minimum non-zero net amount.
- **Operation:** Traverse the list of net amounts to find the minimum.

3.2 **getSimpleMaxIndex()**

- **Purpose:** Find the index of the bank with the maximum net amount, regardless of payment mode compatibility.
- **Operation:** Traverse the list of net amounts to find the maximum.

3.3 **getMaxIndex()**

- **Purpose:** Find the index of the creditor bank that shares a payment mode with the debtor and has the maximum positive balance.
- **Operation:** Compute the intersection of payment modes and track the maximum amount.

3.4 **printAns()**

- **Purpose:** Print the final list of transactions after minimization.
- **Operation:** Output the transactions based on the final answer matrix.

4. Example Workflow

1. **Input:**
    - **For Banks:**
        - Banks: A, B, C, D, E.
        - Payment Modes: t1, t2.
        - Transactions: B owes A 300, C owes A 700, D owes B 500, E owes B 500.
    - **For Personal Transactions:**
        - People and transactions details as provided.

2. **Calculation:**
    - **For Banks:**
        - Compute net balances and minimize transactions through the optimization algorithm.
    - **For Personal Transactions:**
        - Compute net balances and minimize transactions directly.
3. **Output:**
    - **For Banks:** Display transactions required to settle all debts with minimal exchanges.
    - **For Personal Transactions:** Display minimized transactions among individuals.

Conclusion

This project effectively minimizes the number of transactions needed to settle debts among multiple entities by leveraging optimized algorithms and considering constraints such as payment modes for banks. The system ensures that the transactions are minimized to the greatest extent possible, improving efficiency in both personal and banking transactions.