Lab 9 D

Problem 1: Stock Portfolio Management

You are tasked with developing a stock portfolio management system. The system allows users to create an account, manage their stock portfolio, buy and sell shares, and view the value of their portfolio. The system will read commands from a file named "file.txt" and process them accordingly. The commands are as follows:

- **AddUser "username" :** Create a user account with the given username. If the user already exists, print a warning.
- **AddStock "stock_name" "price" :** Add a stock with the given stock name and price. If the stock already exists, print a warning.
- **List :** List all the available stocks with their current prices.
- **Buy "username" "stock_name" "shares" :** Buy the specified number of shares of a stock for the given username. If the user does not exist or the stock does not exist, print a warning.
- **Sell "username" "stock_name" "shares" :** Sell the specified number of shares of a stock for the given username. If the user does not have enough shares or the stock does not exist, print a warning.
- **Portfolio "username" :** View the portfolio for the given user, displaying the stock names, shares owned, and total value.

You also have to implement warnings for the following cases:

- If the user or stock does not exist, print the following: `printf("User %s does not exist.\n", username)` or `printf("Stock %s does not exist.\n", stock_name)`.
- If the user tries to buy or sell more shares than they have, print: `printf("User %s does not have enough shares of %s to sell.\n", username, stock_name)`.

- If the user tries to sell a stock that they have not bought, print: `printf("User %s does not have shares of %s to sell.\n", username, stock_name)`.

Constraints:

- 1 <= number of stocks <= 100
- 1 <= number of users <= 1000
- 1 <= number of commands <= 1000
- 1 <= shares <= 1000
- 1 <= price <= 1e5

Example File:

```
AddUser User1
AddUser User2
AddStock Stock1 50.00
AddStock Stock2 75.00
Buy User1 Stock1 10
Buy User2 Stock2 20
Sell User1 Stock1 5
Portfolio User1
Portfolio User2
List
```

Example Output:

```
User1 bought 10 shares of Stock1.
User2 bought 20 shares of Stock2.
User1 sold 5 shares of Stock1.
Portfolio for User1:
Stock: Stock1, Shares: 5, Total Value: 250.00
```

```
Portfolio for User2:
Stock: Stock2, Shares: 20, Total Value: 1500.00
Stock: Stock1, Price: 50.00
Stock: Stock2, Price: 75.00
```

Problem 2: Movie Rental System

In a movie rental system, you need to manage customers and the collection of movies available for rent. The system should allow the following operations:

- **Add Customer:** Adds a customer to the system with a unique ID and name.
- **Add Movie:** Adds a movie to the rental system with a unique ID, title, director, rental price, and quantity available.
- **Borrow Movie:** Allows a customer to rent a movie, provided it is available in the rental collection.
- **Return Movie:** Allows a customer to return a previously rented movie.
- **Display Movies:** Displays all movies currently available in the rental system with their details (ID, Title, Director, Available Copies, Price).
- **Display Customers:** Displays all customers and their rented movies.

The input file, `movies.txt`, contains a list of operations to be applied to the movie rental system. Each operation will either add a customer, add a movie, allow a customer to rent or return a movie, or display information about the movies and customers.

Each line in the `movies.txt` file follows one of the formats below:

- **Add Customer:** `1 CustomerID CustomerName` – Adds a customer with the specified ID and name.
- **Add Movie:** `2 MovieID MovieTitle Director Price Quantity` – Adds a movie to the rental system with the specified ID, title, director, price, and quantity available.

- **Borrow Movie:** `5 CustomerID Borrow MovieID` – Allows a customer to rent a movie if it is available.
- **Return Movie:** `6 CustomerID Return MovieID` – Allows a customer to return a previously rented movie.
- **Display Movies:** `3` – Shows all movies in the system along with their details.
- **Display Customers:** `4` – Shows all customers and their rented movies.

Task:

1. Use structs to represent customers and movies in the rental system.
2. Maintain a record of each customer's ID, name, rented movies, and the rental system's movies with details such as ID, title, director, price, and quantity available.

Constraints:

- Each line in the file represents a valid operation.
- Number of Operations ≤ 100.
- Movie titles and director names are treated as strings, while customer IDs and movie IDs are integers, and the rental price is a float.

Input Format:

The `movies.txt` file contains lines formatted as follows:

```
1 CustomerID CustomerName
2 MovieID MovieTitle Director Price Quantity
5 CustomerID Borrow MovieID
6 CustomerID Return MovieID
3
4
```

Output Format:

After processing all operations, output the details of all movies in the rental system and the movies rented by each customer in the following format:

```
MovieID MovieTitle Director AvailableCopies Price
```

For customers, output in the following format:

```
CustomerID CustomerName No_of_RentedMovies
```

Structs:

You are required to use structs to model the system. Below are the suggested structures for your implementation (You can Modify them as you wish):

- **Movie Struct:** The `Movie` struct should contain the following information:
  - Movie ID (int)
  - Movie Title (string)
  - Director (string)
  - Price (float, two decimal places)
  - Quantity Available (int)
- **Customer Struct:** The `Customer` struct should contain the following information:
  - Customer ID (int)
  - Customer Name (string)
  - List of Rented Movies (array or dynamic list of Movie IDs)

You may also create additional structs as needed to manage the rental operations efficiently.

Example Input:

```
1 101 John
2 301 Inception ChristopherNolan 5.00 3
2 302 Matrix Wachowski 4.50 2
1 102 Jane
5 101 Borrow 301
5 102 Borrow 302
6 101 Return 301
3
4
```

Example Output:

```
301 Inception ChristopherNolan 3 5.00
302 Matrix Wachowski 1 4.50
101 John 0
102 Jane 1
```

Notes:

- For each operation, maintain a record of the movies in the rental system and the customers rental accounts.