Name:_____ ID: _____

Object-Oriented Programming Lab #10

19/2/2025

Operator Overload & Structs and File Stream

Lab 1: Structure and File Stream - C++ Playhouse Ticket System

Your local community theater is experiencing a tech crisis: their outdated ticket reservation system keeps leading to overbookings. Develop a new command-line ticket reservation system leveraging structs, loops, vectors, and file I/O to manage reservations effectively.

Data Structures

1. **Movie Struct**
   - Create a Movie struct to hold the following fields:
     - title (string): The name of the movie.
     - date (string): The date of the screening in "YYYY-MM-DD" format.
     - availableSeats (2D vector of booleans): A 4x10 grid representing seat availability. Each element corresponds to a seat:
       - true: The seat is available.
       - false: The seat is booked.
2. **Reservation Struct**
   - Create a Reservation struct to hold the following fields:
     - customerName (string): The name of the customer.
     - movieTitle (string): The title of the movie.
     - date (string): The date of the screening.
     - round (int): The round number (1 for 12:00, 2 for 15:00, 3 for 18:00, 4 for 21:00).
     - seatNumber (int): The seat number (1 to 40, where seats are numbered row-wise).

Core Functions

1. displaySchedule(vector<Movie> movies)
   - Takes a vector of Movie structs as input.
   - Displays the schedule neatly, showing the title, date, and round availability.
   - Example output:

```
Captain America Brave New World (2024-02-16)
- Round 1: 10 seats left
- Round 2: 10 seats left
- Round 3: 10 seats left
- Round 4: 10 seats left
Bridget Jones Mad About the Boy (2024-02-17)
- Round 1: 10 seats left
- Round 2: 10 seats left
- Round 3: 10 seats left
- Round 4: 10 seats left
```

2. makeReservation(vector<Movie> &movies)
   - Guides the user through the reservation process:
     - Displays the schedule using displaySchedule.
     - Prompts the user to select a movie, date, and round.

- Checks seat availability and displays a seating chart:
  - Available seats are marked as O.
  - Booked seats are marked as X.
  - Example seating chart:

```
O O O X O O O O O O
O O X X O O O O O O
O O O O O O O O O O
O O O O O O O O O O
```

- Lets the user select an open seat by entering its number (1 to 40).
- Creates a Reservation struct, updates the availableSeats in the corresponding Movie, and confirms the reservation.

3. cancelReservation(vector<Movie> &movies)
   - Allows users to cancel reservations:
     - Prompts the user to enter their name, movie title, date, round, and seat number.
     - Frees up the corresponding seat in the availableSeats vector.
     - Confirms the cancellation.

Main Program & Persistence

- Use a vector<Movie> to hold several movies.
- Start with sample data (5 movies with different dates and availability). For example:

```
Movie movie1 = {"Captain America Brave New World", "2024-02-16",
vector<vector<bool>>(4, vector<bool>(10, true))};
Movie movie2 = {"Bridget Jones Mad About the Boy", "2024-02-17",
vector<vector<bool>>(4, vector<bool>(10, true))};
Movie movie3 = {"Flat Girls", "2024-02-18",
vector<vector<bool>>(4, vector<bool>(10, true))};
Movie movie4 = {"Heretic", "2024-02-19", vector<vector<bool>>(4,
vector<bool>(10, true))};
Movie movie5 = {"Dark Nuns", "2024-02-20",
vector<vector<bool>>(4, vector<bool>(10, true))};
```

- Implement a menu loop with the following options:

  1. **View Schedule** : Call displaySchedule.
  2. **Make Reservation** : Call makeReservation.
  3. **Cancel Reservation** : Call cancelReservation.
  4. **Exit** : Save the current state of the movies to a file (e.g., in CSV or plain text format) and terminate the program.

- Ensure the program loads data from the file when it starts, allowing persistence between runs. For example:
  - Save the data in a structured format like:

```
Movie Title,Date,Round,Seat Number,Availability
Captain America Brave New World,2024-02-16,1,1,true
Captain America Brave New World,2024-02-16,1,2,false
...
```

Additional Notes

1. **Error Handling**
    - Handle invalid input gracefully:
        - If the user enters an invalid round number, display an error message and prompt again.
        - If the user tries to reserve a seat that is already booked, inform them and allow them to choose another seat.
        - If the user cancels a reservation that does not exist, display an appropriate message.
2. **Edge Cases**
    - Test the program with edge cases, such as:
        - All seats booked in a round.
        - Invalid inputs (e.g., non-existent rounds or seat numbers).
        - Attempting to cancel a reservation that does not exist.
3. **File Format**
    - Specify the file format for saving and loading data. For example:
        - Use a CSV file with columns for Movie Title, Date, Round, Seat Number, and Availability.