Programming fundamental

Assignment 1

Do not engage in plagiarism while completing assignments.

Deadline: 17-Dec-2024

1- Theater Seating

You are tasked with designing a C++ program to manage ticket sales in a small theater. The theater has 15 rows, each containing 30 seats. In addition to using 2D arrays and functions, the program should include file handling to save and load data, ensuring persistent storage of seating charts and sales records. Proper input validation must be implemented to ensure robustness.

```
                    Seats
          123456789012345678901234567890
Row  1    ***###***###*########*****####
Row  2    ####*************####*******##
Row  3    **###**********########****###
Row  4    **######**************##******
Row  5    ********#####*********########
Row  6    ##############************####
Row  7    #######************###########
Row  8    ************##****############
Row  9    #########*****############****
Row  10   #####*************############
Row  11   #**********#################**
Row  12   #############********########*
Row  13   ###***********########**######
Row  14   ##############################
Row  15   ##############################
```

Functional Requirements

1. Seat Pricing Setup

- The program should prompt the user to enter ticket prices for each row at the start or load prices from a file.
- Row-specific prices should be stored in a 1D array for later reference.

2. Seating Chart Display

- The seating chart should visually indicate:
  - #: Available seat.
  - *: Sold seat.
- Display row and seat numbers clearly for easy navigation.

3. Sell Tickets

- Allow the user to select a specific row and seat number to sell tickets:
  - Validate that the selected seat is within bounds and available for sale.
  - Calculate the total price of sold tickets based on the row pricing.
  - Update the seating chart to mark seats as sold (*).

4. Sales Statistics

- Track and display key sales metrics:
    - Total revenue generated from ticket sales.
    - Total number of seats sold.
    - Number of available seats in each row.
    - Total number of available seats in the theater.

5. File Handling

- Save the current seating chart and sales data to a file upon program exit.
- Allow the program to load previously saved seating chart and sales data at startup.

6. Menu Options

- Provide the user with the following menu-driven options:
    1. Display the current seating chart.
    2. Sell tickets.
    3. View total sales and seating statistics.
    4. Save current data and exit the program.

Task: 1 **CLO3** **1 Marks**

Do not engage in plagiarism while completing assignments.

Task: 2 **CLO4** **1.5 Marks**

Write and implement comments in your program to explain each function clearly.

Task: 3 **CLO1** **2.5 Marks**

Implement the function initializeSeatingChart to populate a 2D array representing the seating chart. Include comments explaining the purpose and steps of the function. Write a function displaySeatingChart that uses a 2D array to visually display the seating chart. Add detailed comments to describe how the rows and seats are labeled and how the available and sold seats are represented. Implement a function sellTickets that validates user input for row and seat numbers in the 2D seating array. Include comments to explain input validation logic and how the seating chart is updated after a successful sale.

Task: 4 **CLO2** **5 Marks**

Logic Building
This section evaluates your ability to effectively build and implement the logic for managing theater seating and ticket sales. Focus on designing functions that handle tasks such as initializing seating charts, validating user input, updating seat availability, and calculating sales statistics. Proper use of 2D arrays and file handling to store and retrieve data is essential. Demonstrating clear and efficient logic will ensure that your program functions as intended and meets the requirements for a smooth and user-friendly experience.

```
int main() {
    char seating[ROWS][SEATS_PER_ROW];
    double rowPrices[ROWS] = {0};
    double totalRevenue = 0.0;
    int seatsSold = 0;

    string filename = "theater_data.txt";

    // Load data from file or initialize defaults
    if (!loadFromFile(seating, rowPrices, totalRevenue, seatsSold, filename)) {
        initializeSeatingChart(seating);
        inputRowPrices(rowPrices);
    }

    int choice;
    do {
        showMenu();
        cin >> choice;
        switch (choice) {
            case 1:
                displaySeatingChart(seating);
                break;
            case 2:
                sellTickets(seating, rowPrices, totalRevenue, seatsSold);
                break;
            case 3:
                displayStatistics(seating, totalRevenue, seatsSold);
                break;
            case 4:
                saveToFile(seating, rowPrices, totalRevenue, seatsSold, filename);
                cout << "Data saved. Exiting program.\n";
                break;
            default:
                cout << "Invalid choice. Please try again.\n";
        }
    } while (choice != 4);

    return 0;
}
```