**Final Lab Project (200 points):**

**Problem: Theater Ticket Sales**

Create a **TicketManager** class and a program that uses it to sell tickets for a single performance theater production. The project can be implemented as a multifile program, or all the functions can be cut and pasted into a single file. Here are the specifications:

• The theater's auditorium has 15 rows, with 30 seats in each row. To represent the seats, the TicketManager class should have a two-dimensional array of **SeatStructures**. Each of these structures should have data members to keep track of the seat's price and whether it is available or already sold.

• The data for the program is to be read in from given two files The first one, *SeatPrices.dat*, contains 15 values representing the price for each row. All seats in each row are the same price, but different rows have different prices.

The second file, *SeatAvailability.dat*, holds the seat availability information. It contains 450 characters (15 rows with 30 characters each), indicating which seats have been sold ('\*') and which are available ('#'). Initially all seats are available. However, once the program runs and the file is updated, some of the seats will have been sold. The obvious function to read in the data from these files and set up the array is the constructor that runs when the TicketManager

object is first created.

• The client program should be a menu-driven program that provides the user with a menu of box office options, accepts, and validates user inputs, and calls appropriate class functions to carry out desired tasks. The menu should have options to display the seating chart, request tickets, print a sales report, and exit the program.

• When the user selects the display seats menu option, a TicketManager function should be called that creates and returns a string holding a chart, similar to the one shown here. It should indicate which seats are already sold (\*) and which are still available for purchase (#). The client program should then display the string.

**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 ##############################

• When the user selects the request tickets menu option, the program should prompt for the number of seats the patron wants, the desired row number, and the desired starting seat number. A TicketManager ticket request function should then be called and passed this information so that it can handle the ticket request. If any of the requested seats do not exist, or are not available, an appropriate message should be returned to be displayed by the client program. If the seats exist and are available, a string should be created and returned that lists the number of requested seats, the price per seat in the requested row, and the total price for the seats. Then the user program should ask if the patron wishes to purchase these seats.

• If the patron indicates they do want to buy the requested seats, a TicketManager purchase tickets module should be called to handle the actual sale. This module must be able to accept money, ensure that it is sufficient to continue with the sale, and if it is, mark the seat (s) as sold, and create and return a string that includes a ticket for each seat sold (with the correct row, seat number, and price on it).

• When the user selects the sales report menu option, a TicketManager report module should be called. This module must create and return a string holding a report that tells how many seats have been sold, how many are still available, and how much money has been collected so far for the sold seats. Think about how your team will either calculate or collect and store this information so that it will be available when it is needed for the report.

• When the day of ticket sales is over and the quit menu choice is selected, the program needs to be able to write the updated seat availability data back out to the file. The obvious place to do this is in the TicketManager destructor.

**You can work on this project by yourself or with your team partners. Once done, submit source codes and output in a word or pdf file to your lab instructor by due date.**

**Due Date: 4/22/2022 by 11:59 pm**