**Computer Science 1106 Programming Assignment Two --- 50 Points**

Theater Seating (Reduced Ticket Master Application)

Write a program that can be used by a small theater to sell tickets for performances. The theater’s auditorium has 15 rows of seats, with 30 seats in each row. The program should display a screen that shows which seats are available and which are taken.

For example, the following screen shows a chart depicting each seat in the theater. Seats that are taken are represented by an \* symbol, and seats that are available are represented by a # symbol:

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 ##############################

Here is a list of tasks this program must perform:

When the program begins, it should read the seat prices for each row from a file called seatPrice.txt.

The prices can be stored in a separate array. In addition, it should initialize the seating chart from a file indicating which seats have been sold/available for each row.

Once the prices are entered, the program should display a seating chart similar to the one shown above. The user may enter the row and seat numbers for tickets being sold. Every time a ticket or group of tickets is purchased, the program should display the total ticket prices and update the seating chart.

The program should keep a total of all ticket sales. The user should be given an option of viewing this amount.

The program should also give the user the option to see a list of how many seats have been sold, how many seats are available in each row, and how many seats are available in the entire auditorium.

***Input Validation:*** *When tickets are being sold, do not accept row or seat numbers that do not exist. When someone requests a particular seat, the program should make sure that seat is available before it is sold.*

**Due Date:**

* As indicated on the drop box folder: Programming Assignment Two

**What to turn in:**

1. Copy and paste your source code into a word document and paste screen shots of the execution of your program directly under your source code in the document.
2. A hard copies of the source code for the program.
3. Compress .cpp file, .txt , and word files into **a single file** with the filename: yournameProgram2.zip
4. Place the compressed file into the D2L drop box called Programming Assignment Two.
5. Hand in the hard copies of the programs and the word document at the beginning of class on the due date indicated by the folder in the D2L drop box.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Programmer: Chris Dang Class: CSCI 1106 Fall 2014

//

// Description: This is a reduced ticket master program designed to allow the

// purchase of tickets.

// |

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <iostream>

#include <iomanip> // for ticket prices

#include <fstream> // for reading in prices & seat chart and write to chart

#include <string>

using namespace std;

const int ROW\_SIZE = 15 ; // number of seats per row

const int COL\_SIZE = 30 ; // number of rows

const char EMPTY\_SEAT\_ICON = '#' ;

const char SOLD\_SEAT\_ICON = '\*' ;

const string TITLE = "Reduced Ticket Master Application" ;

const string MENU\_HEADER = "Select an option:" ;

const string MENU1 = "1: View Seating Chart" ;

const string MENU2 = "2: Purchase Tickets" ;

const string MENU3 = "3: View Total Ticket Sales" ;

const string MENU4 = "4: View Seating Info (Number of sold, and open seats per row and total)" ;

const string MENU5 = "0: Exit Program" ;

const string MENU\_SELECTION = "Enter selection : " ;

const string SEAT\_NUMBERS = "123456789012345678901234567890" ;

const string TICKET\_HEADER = "How many tickets would you like to purchase?" ;

const string ROW\_HEADER = "Which row would you like your seat? (1-15)" ;

const string COL\_HEADER = "Which seat nuumber would you like? (1-30)" ;

const string ROW\_SELECTION = "Enter row: " ;

const string COL\_SELECTION = "Enter seat number: " ;

void getTicketPrices(double seatPrices[]) ;

void createSeatChart(bool seats[ROW\_SIZE][COL\_SIZE]) ;

void getSeatChart(bool seats[ROW\_SIZE][COL\_SIZE]) ;

void displayMenu() ;

void displaySeatChart(bool seats[ROW\_SIZE][COL\_SIZE]) ;

int getMenuChoice() ;

int getNumOfTickets() ;

int getRow() ;

int getSeatNum() ;

bool checkIfRowSold (bool seats[ROW\_SIZE][COL\_SIZE], int getRow) ;

void purchaseTicketAndUpdateCharts(bool seats[ROW\_SIZE][COL\_SIZE], double seatPrices[]) ;

void displayTotalTicketSales (bool seats[ROW\_SIZE][COL\_SIZE], double seatPrices[]) ;

void displaySeatInfo (bool seats[ROW\_SIZE][COL\_SIZE]) ;

void exitMessage () ;

int main () {

//declaration

bool seats[ROW\_SIZE][COL\_SIZE];

double seatPrices[ROW\_SIZE];

//initial startup

getTicketPrices(seatPrices) ;

getSeatChart(seats) ;

//console output starts

displaySeatChart(seats) ;

cout << TITLE << endl << endl ;

//menu displays in console

//loops repeats until sentinal value is used//////////////////////////////

int userChoice = getMenuChoice() ;

while (userChoice != 0) {

switch(userChoice){

case 1: displaySeatChart(seats) ;

break ;

case 2: purchaseTicketAndUpdateCharts(seats, seatPrices) ;

break ;

case 3: displayTotalTicketSales (seats, seatPrices) ;

break ;

case 4: displaySeatInfo (seats) ;

break ;

} // end switch

userChoice = getMenuChoice() ;

}// end while

// end loop //////////////////////////////////////////////////////////

exitMessage() ;

return 0;

}// end main

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Function: getTicketPrices

//

// Description: Reads in ticket prices from text file and enters them into array

//

// Pre: seatPrice.txt must be in source folder to be read.

// The array that contains the prices must be declared beforehand.

//

//

// Post: Returns false if read correctly.

// Returns true if problem occurs.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void getTicketPrices(double seatPrices[ROW\_SIZE]) {

ifstream infile ;

infile.open("seatPrice.txt") ;

if(!infile){

cerr << "Error opening file. --- Terminating Program. " << endl <<

endl ;

exit (1106);

} // end if

for (int i = 0; i < ROW\_SIZE; i++) {

double price = 0 ;

infile >> price ;

seatPrices[i] = price ;

} // end for

}// end getTicketPrices

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Function: createSeatChart

//

// Description: Function will create an empty seat chart.

//

// Pre: Two global int constants (ROW\_SIZE and COL\_SIZE) for the

// size of the chart must be initialized. Global const char EMPTY\_SEAT\_ICON

// must be initialized as well.

// fstream library is needs to be included.

//

// Post: Returns nothing

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void createSeatChart() {

ofstream outfile;

outfile.open("seatChart.txt");

if(!outfile){

cerr << "Error writing file. --- Terminating Program. " << endl <<

endl;

exit (1106);

} // end if

for (int i = 0; i < ROW\_SIZE; i++){

for (int j = 0; j < COL\_SIZE; j++) {

outfile << EMPTY\_SEAT\_ICON ;

}// end for

outfile << endl;

}// end for

}// end createSeatChart

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Function: createSeatChart

//

// Description: Function will check for seatChart.txt. If the chart isn't found

// it will call createSeatChart and pull from the new chart.

//

// Pre: Two global int constants (ROW\_SIZE and COL\_SIZE) for the

// size of the chart must be initialized. Global const char EMPTY\_SEAT\_ICON

// must be initialized as well.

// fstream library is needs to be included.

// A 2 Dimension bool array must be declared in main.

//

// Post: Returns nothing

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void getSeatChart(bool seats[ROW\_SIZE][COL\_SIZE]) {

char seat;

ifstream infile;

infile.open("seatChart.txt");

if(!infile){

createSeatChart() ;

infile.open("seatChart.txt");

cout << "I open bad\n" ;

} // end if

for (int i = 0; i < ROW\_SIZE; i++) {

for (int j = 0; j < COL\_SIZE; j++) {

infile >> seat ;

if (seat == EMPTY\_SEAT\_ICON)

seats[i][j] = false;

else if (seat == SOLD\_SEAT\_ICON)

seats[i][j] = true;

} //end for

} // end for

} // end getSeatChart

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Function: displayMenu

//

// Description: Function will display a menu with a header and four menu options

//

// Pre: Global string constants for the MENU\_HEADER, MENU1, MENU2, MENU3, MENU4

// must be intialized beforehand.

//

// Post: Returns nothing

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void displayMenu() {

cout << MENU\_HEADER << endl <<

MENU1 << endl <<

MENU2 << endl <<

MENU3 << endl <<

MENU4 << endl <<

MENU5 << endl << endl << MENU\_SELECTION ;

}// end displayMenu

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Function: getMenuChoice

//

// Description: Function will get the menu choice from the user

//

// Pre: displayMenu function must be able to be called.

//

// Post: Returns choice as an integer

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int getMenuChoice() {

int choice ;

displayMenu() ;

cin >> choice ;

while (choice != 1 && choice != 2 && choice != 3 && choice != 4 && choice != 0){

cout << "Error:\n" ;

cout << "Please enter menu choice 1, 2, 3, or 4\n" ;

displayMenu() ;

cin >> choice ;

} // end while

return choice ;

}// end getMenuChoice

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Function: displaySeatChart

//

// Description: Function will display the seat chart from

// 2 Dimension array seats

//

// Pre: 2-Dimension bool array seats must be initialized beforehand.

// global char constants EMPTY\_SEAT\_ICON and SOLD\_SEAT\_ICON must be

// initialized.

//

// Post: Returns nothing

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void displaySeatChart(bool seats[ROW\_SIZE][COL\_SIZE]) {

cout << "Seating Chart\n" << setw(37) << SEAT\_NUMBERS << endl;

for (int i = 0; i < ROW\_SIZE; i++) {

cout << "Row " << setw(2) << (i + 1) << setw(2); // displays Row number

for (int j = 0; j < COL\_SIZE; j++) {

if (seats[i][j] == false)

cout << EMPTY\_SEAT\_ICON ;

else if (seats[i][j] == true)

cout << SOLD\_SEAT\_ICON ;

} // end for

cout << endl ; // output formatting

}// end for

cout << endl << endl ; // output formatting

} // end displaySeatChart

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Function: getNumOfTickets

//

// Description: Function will get the number of tickets they want to purchase

// from the user

//

// Pre: const string TICKET\_HEADER must be initialized.

//

// Post: Returns int tickets

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int getNumOfTickets() {

int tickets ;

cout << TICKET\_HEADER << endl

<< "Tickets: " ;

cin >> tickets ;

while (tickets < 1) { //makes sure user enters more than 1 ticket

cout << "\nError: Please request 1 or more tickets\n\n" <<

TICKET\_HEADER << endl <<

"Tickets: " ;

cin >> tickets ;

}// end while

return tickets ;

} // end getNumOfTickets

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Function: getRow

//

// Description: Function will get the row number for the seat the user wants

//

// Pre: const string ROW\_HEADER must be initialized.

// const string ROW\_SELECTION must be initialized.

//

// Post: Returns int row

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int getRow() {

int row ;

cout << ROW\_HEADER << endl

<< ROW\_SELECTION ;

cin >> row ;

while (row < 1 || row > ROW\_SIZE) {

cout << "\nError: Row does not exist\n\n" <<

ROW\_HEADER << endl <<

ROW\_SELECTION ;

cin >> row ;

}// end while

return row ;

}// end getRow

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Function: getSeatNum

//

// Description: Function will get the seat number for the seat the user wants

//

// Pre: const string COL\_HEADER must be initialized.

// const string COL\_SELECTION must be initialized.

//

// Post: Returns int row

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int getSeatNum() {

int seatNum ;

cout << COL\_HEADER << endl

<< COL\_SELECTION ;

cin >> seatNum ;

while (seatNum < 1 || seatNum > COL\_SIZE) {

cout << "\nError: Seat number does not exist\n\n" <<

COL\_HEADER << endl <<

COL\_SELECTION ;

cin >> seatNum ;

}// end while

return seatNum ;

}// end getRow

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Function: storeSeatChoice

//

// Description: Function will stire the seat number for the seat the user wants

//

// Pre: must be able to use getSeatNum function

// 2-Dimension bool array seats[][] must be initialzed

// getSeatNum must have been used

// getRow must have been used

//

// Post: Returns nothing

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void storeSeatChoice(int row, int seatNum, bool seats[ROW\_SIZE][COL\_SIZE]) {

seats[row][seatNum] = true ;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Function: checkIfSeatSold

//

// Description: Function will check if the seat has been sold

//

// Pre: 2-Dimension bool array seats[][] must be initialized and passed in

// getSeatNum must be able to be used

// getRow must be able to be used

// const int ROW\_SIZE must be initialized

// const int COL\_SIZE must be initialized

//

// Post: Returns true if seat has been sold. Returns false if open.

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

bool checkIfSeatSold(int getRow, int getSeatNum, bool seats[ROW\_SIZE][COL\_SIZE]){

if (seats[getRow-1][getSeatNum-1] == true) {

cout << "\nSeat has been sold. It is not available for purchase\n\n" ;

return true ;

}// end if

return false ;

}// end checkIfSeatSold

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Function: checkIfRowSold

//

// Description: scans row in 2-Dimension array to see if all seats are sold.

// If all seats are sold, function will return true.

//

// Pre: 2-Dimension bool array must be passed in and initialized

// getRow must be used

//

// Post: Returns true if row is sold out. Returns false otehrwise

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

bool checkIfRowSold (bool seats[ROW\_SIZE][COL\_SIZE], int getRow) {

int seatCount = 0 ;

for (int i = 0; i < COL\_SIZE; i++)

if (seats[getRow-1][i] == true)

seatCount++;

if (seatCount == COL\_SIZE){

cout << "\nRow has been sold. It is not available for purchase\n\n" ;

return true ;

}// end if

else return false ;

}// end checkIfRowSold

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Function: purchaseTicket

//

// Description: Allows the purchase of tickets from theatre. Charts are updated

// in function to prevent seats from being double booked

//

// Pre: 2-Dimension bool array must be passed in and initialized

// seatPrices must be declared and initialized

//

// Post: Returns nothing

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void purchaseTicketAndUpdateCharts(bool seats[ROW\_SIZE][COL\_SIZE], double seatPrices[]) {

int tickets = getNumOfTickets() ;

double total = 0;

// holds prices for the tickets that the user bought

int \*ticketPrices = new int [tickets] ;

if(ticketPrices == NULL) {

cout << "\n\n Error allocating memory --- Terminating\n\n" ;

exit(1106) ;

}// end if

for (int i = 0; i < tickets; i++) {

int row = getRow() ;

while (checkIfRowSold(seats, row) == true)

row = getRow() ;

int seatNum = getSeatNum() ;

while (checkIfSeatSold(row, seatNum, seats) == true)

seatNum = getSeatNum() ;

ticketPrices[i] = seatPrices[row-1] ;

// seat is sold, therefore that seat is set to true for sold

seats[row-1][seatNum-1] = true ;

// writing the sold seat to file///////////////////////////////////

ofstream outfile;

outfile.open("seatChart.txt");

if(!outfile){

cerr << "Error writing file. --- Terminating Program. " << endl <<

endl;

exit (1106);

} // end if

for (int i = 0; i < ROW\_SIZE; i++) {

for (int j = 0; j < COL\_SIZE; j++) {

if (seats[i][j] == false)

outfile << EMPTY\_SEAT\_ICON ;

else if (seats[i][j] == true)

outfile << SOLD\_SEAT\_ICON ;

} // end for

outfile << endl ; // output formatting

}// end for

//end writing seat to file //////////////////////////////////////////

} // end for

for (int i = 0; i < tickets; i++)

total = total + \*(ticketPrices + i) ;

cout << "The total for the tickets is: $" << total << endl << endl ;

}// end purchaseTicket

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Function: displayTotalTicketSales

//

// Description: Display the total ticket sales for the theatre

//

// Pre: The parameters are:

// 2-Dimension bool seats must be initialized and passed in.

// double seatPrices array must be initilized and passed in.

//

// Post: Returns nothing

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void displayTotalTicketSales (bool seats[ROW\_SIZE][COL\_SIZE], double seatPrices[]) {

double total = 0 ;

for (int i = 0; i < ROW\_SIZE; i++)

for (int j = 0; j < COL\_SIZE; j++)

if (seats[i][j] == true)

total = total + seatPrices[i] ;

cout << "\nThe theatre's total ticket sales is: $" << total << endl << endl;

}// end displayTotalTicketSales

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Function: displaySeatInfo

//

// Description: Display seating info for the theatre

// Info includes:

// Total seats available

// The number of seats available per row

// The number of seats sold throught the theatre

//

// Pre: The parameters are:

// 2-Dimension bool seats must be initialized and passed in.

//

// Post: Returns nothing

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void displaySeatInfo (bool seats[ROW\_SIZE][COL\_SIZE]) {

int theatreTotal = 0, totalSold = 0;

//Display seats available per row

cout << "\nListed below is the number of seats avaiable per row.\n" ;

for (int i = 0; i < ROW\_SIZE; i++) {

int rowTotal = 0 ;

cout << "Row " << setw(2) << (i + 1) << ": "; // displays Row number

for (int j = 0; j < COL\_SIZE; j++) {

if (seats[i][j] == false)

rowTotal++ ;

} // end for

cout << rowTotal << " seats available." << endl ;

theatreTotal = theatreTotal + rowTotal ;

}// end for

//Display total seats available

cout << "\nThe total number of seats available is: " << theatreTotal << endl;

//Display number of seats sold

// ROW\_SIZE \* COL\_SIZE gives total number of seats in the theatre.

// Total seats - seats avaiable = seats sold

totalSold = (ROW\_SIZE \* COL\_SIZE) - theatreTotal ;

cout << "\nThe total number of seats sold is: " << totalSold<< endl;

cout << endl << endl ;

} // end displaySeatInfo

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Function: exitMessage

//

// Description: Message to indicate program is closing

//

// Pre: no parameters

//

// Post: Returns nothing

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void exitMessage () {

cout << "\n\nTicket Master Closing\n\n" ;

}// end exitMessage ()





