2021F CS234 Computer Science II

Lab 4
Total points: 100

P6.7 A theater seating chart is implemented as a **two-dimensional array** of ticket prices, like this:

```
10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10
10 10 20 20 20 20 20 20 10 10
10 10 20 20 20 20 20 20 10 10
10 10 20 20 20 20 20 20 10 10
20 20 30 30 40 40 30 30 20 20
20 30 30 40 50 50 40 30 30 20
30 40 50 50 50 50 50 50 40 30
```

Write a **program** that **presents** to the user the above array of **ticket prices.**

Then, the program needs to **ask** the user to pick **either** a **seat**, a **price**, or to **quit** the program.
Your program needs to **validate** that the user selects the **correct** option. You can use **s** for seat, **p** for price, and **q** for quit.
If the user writes a different option, your program needs to **keep asking** for a valid one.

If the user selects the option **(q)uit**, then the program must **terminate.**

If the user selects the option **(p)rice**, then the program needs to **ask** for a price value. (Optional: You can validate the price)
After that, the program must **select** any **available** seat of that price and **sell** it.
If there is **no available** seat of that price, the program must **output** **"Sorry, no seat found with that price."**

If the user selects the option **(s)eat**, then the program needs to **ask** for the **row** and **seat** number. The program must **validate** that the row and seat number are **valid** and **output** a proper **message** to the user. Rows and seat numbers **start from 1.**
After that, the program must **sell** the seat only **if** it is **available.**
If the seat is **already occupied**, the program must output **"Sorry, seat already occupied".**

Mark sold seats by **changing** the price to **0.**

After any **transaction**, the program must **show** the array of **ticket prices** (showing the seats already sold). And the program must **show** the user the **options** to choose from in case the user wants to buy another ticket.

Your program **must implement** the following methods (besides the main method):

```
/**
      Prints the price of seats in a grid like pattern.
      @param seats a 2D array of prices
*/
public static void printSeats(int[][] seats)


/**
      Marks a seat with the price given to 0.
      @param seats the array of seat prices
      @param price the price to mark to zero
*/
public static void sellSeatByPrice(int[][] seats, int price)


/**
      Marks a seat based on a given row and seat number from input.
      @param seats the array of seat prices
*/
public static void sellSeatByNumber(int[][] seats)
```

You can create any additional methods if you need.

Submission details:

Upload a **single ZIP** file.
Name your file as follows: **Lab4_Lastname_Firstname.zip**
There is a **10% points deduction** if your file does not have the correct name.

Your .zip file must contain the following:

1. Your **.java** source files (no .class).
2. A **SINGLE PDF** with screenshots from your programs running (10% points deduction if you don't submit a SINGLE PDF file)
3. A **.txt file** (i.e., readme.txt) with the instructions on how to compile and execute your program **using the command line interface** (i.e., javac and java).

In each .java file, **write** as a multiline comment at the beginning of the file the following:

1. Your name
2. The ID of the problem (e.g., P6.7)
3. The course section

The **zip** file must be uploaded to Canvas. I do not accept answers via email.
I do not accept image files; it must be a PDF file.

Make sure to check the **due date** for this activity on Canvas.
Make sure you are **submitting the correct files**. I will grade the file uploaded to Canvas.

Make sure you test your program using the *javac* and *java* commands before submitting your solution. Follow your own instructions in the .txt file. Make sure to review the grading rubric.
Late submissions are not allowed.