SINGAPORE UNIVERSITY OF SOCIAL SCIENCES

ICT 133
Structured Programming

Tutor-Marked Assignment

July 2022 Presentation

TUTOR-MARKED ASSIGNMENT (TMA)

This assignment is worth 24 % of the final mark for ICT133, Structured Programming.

The cut-off date for this assignment is **Wednesday, 07 September 2022, 2355hrs.**

Assignment Requirements:

- Do NOT define classes for this TMA.
- **Unless specified in the question**, **you CANNOT use packages not covered in this module, e.g., re, collections, numpy, pandas etc.**
- All functions must be documented. Provide sufficient comments to your code and ensure that your program adheres to good programming practices such as not using global variables.
- **Do not use the exit() function.**
- Failing to do so can incur a penalty of as much as 50% of the mark allotted.

Submission Details:

- Use the template word document provided - **SUSS_PI_No-FullName_TMA.docx.** Rename the file with your SUSS PI and full name join with "_TMA" e.g., "**SUSS_PI-TomTanKinMeng_TMA.docx**" (without the quotes).
- Include the following particulars on the first page of the word document, on separate lines: **Course Code, SUSS PI No., Your Name, Tutorial Group and Submission Date**.
- Copy and paste the source code of each program you write in **text** format. Submit screenshots for **only** output of your program, where applicable.
- If you submit the source code as a screenshot, your code will not be marked. That is, **screenshot code will be awarded zero mark.**
- Submit your solution in the form of a **single MS Word document. Do NOT submit as a pdf document**. You will be penalised if you fail to submit a word document.
- The word document must be submitted to your respective T group. Besides this, you are required to upload your source code to Vocareum.

Answer all questions. (Total 100 marks)

Question 1 (20 marks)

This question covers materials in Seminar 1 and 2. Use and express selection structure for this question. Functions, repetition, or collections are not required for this question.

Company O uses the following table to help customers determine their T-Shirt size.

All measurements are in centimetres.

	1	2	3	4	5	6	7	8
Chest	less than 80	80-88	88-96	96-104	104-112	112-120	120-128	128 or more

Table 1-1

(a) Write a program that helps customers find their T-Shirt size. The program reads in the chest measurement in centimetres and displays the T-Shirt size.

Assume user will enter valid input. Sample program executions are as follows:

Sample 1	**Comments**
Enter chest measurement (cm): **80** Your T-Shirt size is 2	Chest measurement 80cm up to 88cm is size 2.
Sample 3	
Enter chest measurement (cm): **104** Your T-Shirt size is 5	Chest measurement 104cm up to 112cm is size 5.
Sample 4	
Enter chest measurement (cm): **70.5** Your T-Shirt size is 1	For chest measurement lesser than 72cm, recommended size is 1.
Sample 5	
Enter chest measurement (cm): **129** Your T-Shirt size is 8	For chest measurement 128cm or more, recommended size is 8.

(6 marks)

As online ordering has increased in folds during the pandemic, the number of returns and exchanges also increased, due to wrong fittings. In order to provide better services and better fittings, company O has enhanced Table 1-1 to include Height and Waist measurements.

All measurements are in centimetres.

	1	2	3	4	5	6	7	8
Height	less than 155	155-160	160-165	165-170	170-175	175-180	180-185	185 or more
Chest	less than 80	80-88	88-96	96-104	104-112	112-120	120-128	128 or more
Waist	less than 70	70-76	76-84	84-92	92-100	100-108	108-116	116 or more

Table 1-2

(b) Using Table 1-2, write a program to recommend the T-Shirt size, based on customers' height, chest, and waist measurements. Your program must display the appropriate "fitting" based on the following fitting rules:

- best fit: if all three measurements indicate same size.
- regular fit: if two measurements indicate same larger size than the third. See Sample execution 2 below.
- relaxed fit: if one measurement indicates larger size than the other two. See sample execution 3 below.

Assume user will enter valid input.

Sample program executions are as follows:

Sample 1	**Comments**
Enter height measurement (cm): **174** Enter chest measurement (cm): **105** Enter waist measurement (cm): **98** You are size 5 (best fit)	All 3 measurements indicated size 5.
Sample 2	
Enter height measurement (cm): **164** Enter chest measurement (cm): **102** Enter waist measurement (cm): **90** You are size 4 (regular fit)	Height measurement 164cm is size 3. Chest measurement 102cm is size 4. Waist measurement 90cm is size 4.
Sample 3	
Enter height measurement (cm): **174** Enter chest measurement (cm): **95** Enter waist measurement (cm): **85** You are size 5 (relaxed fit)	Height measurement 174cm is size 5. Chest measurement 95cm is size 3. Waist measurement 85cm is size 4.

(14 marks)

For each part, submit separate python codes and paste screenshots of at least **THREE (3)** program executions, with different input.

Question 2 (25 marks)

This question covers materials up to Seminar 3. Make use of functions, selection, and repetition structures. NO data structures like sets, lists or dictionary should be used for this question. Keep the program modular by defining other functions if necessary.

Scissors-Paper-Stone (also known by other orderings of the three items, with "stone" sometimes being called "rock") is a hand game originating from China, usually played between two people, in which each player simultaneously forms one of three shapes with an outstretched hand. These shapes are "stone" (a closed fist), "paper" (a flat hand), and "scissors" (a fist with the index finger and middle finger extended, forming a V).

The game has three outcomes: a draw, a win or a loss. A player who decides to play stone will beat another player who has chosen scissors ("stone crushes scissors" or "breaks scissors") but will lose to one who has played paper ("paper covers stone"); a play of paper will lose to a play of scissors ("scissors cuts paper"). If both players choose the same shape, the game is tied (draw). [extracted from Wikipedia, the free encyclopaedia]

(a) Implement **TWO (2)** functions:

- `getShape()` that prompts, validates and returns the player's shape selection of "Scissors", "Paper", "Stone" in uppercase.
- `getRandomShape()` that returns a random pick from "Scissors", "Paper", "Stone" in uppercase.

A sample run of function `getShape()` and `getRandomShape()` as follows:

	Comments
	Execution from print(getShape())
please select a shape: **tone** Sorry, please select from Scissors, Paper, Stone please select a shape: **stone**	Validate the user input Accept uppercase or lowercase of shapes
STONE	Output from print(getShape())
SCISSORS	Output from print(getRandomShape())

(5 marks)

(b) Write a program to play the game of Scissors-Paper-Stone, using structured programming. Your program will allow a player to play three rounds of Scissors-Paper-Stone against the computer.

The scope of the program is as follows:

- At start of the game, allow player to enter his/her name.
- For each round, computer will randomly pick a shape, and ask player to select a shape. Make use of the functions you created in question 2(a).
- Your program will compare the player's shape against the computer's pick, determine the outcome of this round and award one point to the winner. Do not award point if it is a tie.
- Display current score before start of next round.
- At the end of three rounds, display the winner of the game (if any).
- If the result after 3 rounds is a tie, repeat the game (3-rounds) again, until a winner can be determined.

A sample program execution is as follows:

	Comments
Enter player's name: **Alan** Round 1: Alan, please select a shape: **Stone** Round 1: Computer's shape is: PAPER << Alan 0 : Computer 1 >> Round 2: Alan, please select a shape: **STONE** Round 2: Computer's shape is: SCISSORS << Alan 1 : Computer 1 >> Round 3: Alan, please select a shape: **paper** Round 3: Computer's shape is: PAPER << Alan 1 : Computer 1 >>	Award 1 point to computer for winning round 1 Award 1 point to player for winning round 2 No point to be awarded since same shapes Need to play another 3 rounds since Alan and computer both has 1 point
It's a tie!! Rematch... Round 1: Alan, please select a shape: **Stone** Round 1: Computer's shape is: PAPER << Alan 0 : Computer 1 >> Round 2: Alan, please select a shape: **STONE** Round 2: Computer's shape is: STONE << Alan 0 : Computer 1 >> Round 3: Alan, please select a shape: **paper** Round 3: Computer's shape is: PAPER << Alan 0 : Computer 1 >>	
Computer is the winner!!	Program ends once a winner is determined

(10 marks)

(c) Write a new version of the Scissors-Paper-Stone game that will need a player to win **THREE (3)** consecutive rounds to be winner.

The scope of the program is as follows:

- At start of the game, allows player to enter his/her name.
- For each round, computer will randomly pick a shape, and ask player to select a shape. Make use of the functions you created in question 2(a).
- Your program will compare the player's shape against the computer's pick, determine the outcome of this round:
  - Award one point to the winner. Reset the loser's score to 0.
  - Do not award point if it is a tie. Reset both player and computer score to 0
- Repeat the round until a player reaches THREE points (i.e., wins 3 consecutive rounds)

A sample program execution is as follows:

	Comments
Enter player's name: **Alan** Round 1: Alan, please select a shape: **Stone** Round 1: Computer's shape is: PAPER << Alan 0 : Computer 1 >> Round 2: Alan, please select a shape: **STONE** Round 2: Computer's shape is: SCISSORS << Alan 1 : Computer 0 >> Round 3: Alan, please select a shape: **paper** Round 3: Computer's shape is: PAPER << Alan 0 : Computer 0 >> Round 4: Alan, please select a shape: **Stone** Round 4: Computer's shape is: PAPER << Alan 0 : Computer 1 >> Round 5: Alan, please select a shape: **STONE** Round 5: Computer's shape is: PAPER << Alan 0 : Computer 2 >> Round 6: Alan, please select a shape: **scissors** Round 6: Computer's shape is: STONE << Alan 0 : Computer 3 >> Computer is the winner after 6 rounds!!	Award 1 point to computer and reset player's score Award 1 point to player and reset computer' score Reset both scores since same shapes Award 1 point to computer and reset player's score Award 1 point to computer and reset player's score Award 1 point to computer and reset player's score Program ends once a winner is determined

(10 marks)

Question 3 (25 marks)

This question covers materials up to seminar 4. The data structure to use is List. You can use more than ONE list. No nested list or dictionary collection is required for this question.

This question is similar to Q2, but with Scissors-Paper-Stone played like poker game.

(a) Implement a function `getHandOfShapes(size, auto)` that has an integer parameter (size) representing the number of shapes and a boolean parameter (auto) indicating if the shapes are randomly selected. Size must be at least 3. If auto is False, the function will prompt the player to indicate the shapes (Scissors-Paper-Stone) they would like to have, in sequential order and returns the selected shapes as a list.

Duplicated shapes are fine. Examples for a hand of shapes (size 6):

- SCISSORS, PAPER, STONE, PAPER, STONE, SCISSORS
- PAPER, PAPER, PAPER, PAPER, PAPER, PAPER

A sample run of this function `print( getHandOfShapes(4, False) )` is as follows:

	Comments
Shape 1: please select a shape: **scissors** Shape 2: please select a shape: **PAPER** Shape 3: please select a shape: **scissors** Shape 4: please select a shape: **Stone** ['SCISSORS', 'PAPER', 'SCISSORS', 'STONE']	Execution from getHandOfShapes(4, False) Output from the print() statement
['STONE', 'STONE', 'SCISSORS', 'PAPER']	Output from print(getHandOfShapes(4, True)) where the 4 shapes are randomly selected

(5 marks)

(b) Employ structured programming to develop a new game with rules as follows:

- At start of the game, prompt for the size of hand.
- Size must be at least 3. If not, ask user to re-enter the size.
- After capturing the player's name, proceed to create the player's hand of shapes. Make use of the `getHandOfShapes()` function in Q3(a).
- Your program will also setup computer's hand of shapes, according to the size entered.
- Next is to compare the computer and players' hand of shapes, one-by-one, and award one point to the winner. Do not award point if it is a tie.
- After comparing the hand of shapes, display the winner of the game (if any).
- If the result is a tie, repeat the game with new hand of shapes, until a winner can be determined.

Sample program executions are as follows:

Sample 1	**Comments**
Enter size of hand: **4** Enter player's name: **Alan** Shape 1: please select a shape: **scissors** Shape 2: please select a shape: **PAPER** Shape 3: please select a shape: **scissors** Shape 4: please select a shape: **Stone** Game starts... Round 1: Alan SCISSORS : Computer PAPER << Alan 1 : Computer 0 >> Press <Enter> to proceed Round 2: Alan PAPER : Computer PAPER << Alan 1 : Computer 0 >> Press <Enter> to proceed Round 3: Alan SCISSORS : Computer STONE << Alan 1 : Computer 1 >> Press <Enter> to proceed Round 4: Alan STONE : Computer SCISSORS << Alan 2 : Computer 1 >> It's a tie!! Rematch... Shape 1: please select a shape: **scissors** Shape 2: please select a shape: **scissors** Shape 3: please select a shape: **PAPER** Shape 4: please select a shape: **Stone** Game starts... Round 1: Alan SCISSORS : Computer PAPER << Alan 1 : Computer 0 >> Press <Enter> to proceed Round 2: Alan SCISSORS: Computer PAPER	Accept uppercase or lowercase of shapes Award 1 point to player for winning this round Not to award point since same shapes Award 1 point to computer for winning this round Award 1 point to player for winning this round Need to play again!! Award 1 point to player for winning this round

<< Alan 2 : Computer 0 >> Press <Enter> to proceed Round 3: Alan PAPER: Computer STONE << Alan 2 : Computer 1 >> Press <Enter> to proceed Round 4: Alan STONE : Computer PAPER << Alan 3 : Computer 1 >> Alan is the winner!!	Award 1 point to player for winning this round Award 1 point to computer for winning this round Award 1 point to player for winning this round Program ends once a winner is determined

(10 marks)

(c) Write a new game, by enhancing Q3(b) with the following new rules:

- In a hand of `n` shapes, there should not be more than `n/2` of same shapes. This applies to both player and computer.
- After comparing the hand of shapes and the result is a tie, the game will into a playoff.
- In a playoff, computer will randomly pick a shape, and ask player to select a shape. Make use of the functions you created in Q2(a).
- Your program will compare the player's shape against the computer's pick and display the winner. If it is still a tie, this playoff format continues until there is a winner.

A sample program execution is as follows:

	Comments
Enter size of hand: **4** Enter player's name: **Alan** Shape 1: please select a shape: **scissors** Shape 2: please select a shape: **SCISSORS** Shape 3: please select a shape: **scissors** Cannot have more than 2 SCISSORS!! Shape 3: please select a shape: **Paper** Shape 4: please select a shape: **Stone** Game starts... Round 1: Alan SCISSORS : Computer PAPER << Alan 1 : Computer 0 >> Press <Enter> to proceed Round 2: Alan SCISSORS: Computer STONE << Alan 1 : Computer 1 >> Press <Enter> to proceed Round 3: Alan PAPER : Computer SCISSORS << Alan 1 : Computer 2 >> Press <Enter> to proceed Round 4: Alan STONE : Computer SCISSORS << Alan 2 : Computer 2 >> It's a tie!! PLAYOFF... Playoff 1: Alan, please select a shape: **Stone** Playoff 1: Computer's shape is: STONE Playoff 2: Alan, please select a shape: **STONE** Playoff 2: Computer's shape is: SCISSORS Alan is the winner!!	In a hand of 4, cannot have more than 2 of the same shape Award 1 point to player for winning this round Award 1 point to computer for winning this round Award 1 point to computer for winning this round Award 1 point to player for winning this round Playoff time... Program ends once a winner is determined

(10 marks)

Submit separate python codes for each part. Paste screenshot of a program executions that covers all scenarios.

Question 4 (30 marks)

The question covers concepts in all the seminars. Employ structure programming and use of functions to make the program modular.

FR (Fry Rice) is a popular art theatre company and is looking for a simple application to help them dynamically setup seating arrangements for their stage performances or productions. This is important to FR due to the different measures applied during the pandemic. In addition, FR is also looking for the application to allow FR admin to perform booking and cancellation of seats.

Apply data structures to store and process information. The scope and assumptions for this question are as follow:

- Each performance has its own seating plan.
- To setup a performance, FR uses a file to store the seating plan, with filename consisting of production title and performance datetime. For example,
  - Production title "Bad Citizen", performing on 18-Oct-2022 19:30 will have the seating plan setup in file `Bad Citizen-202210181930.txt`
  - Production title "Monkey Goes East", performing on 8-Sep-2022 14:30 will have the seating plan setup in file `Monkey Goes East-202209081430.txt`
  - See Appendix A for more files for different productions' seating plans.
- Blocked seats are marked with #, empty or available seats are marked with O, while those booked are marked with X.

(a) A theatre has rows of seats. The rows are labelled A, B, C, etc and seat numbers 1, 2, 3 etc. A seating plan for production title "Monkey Goes East", performing on 8-Sep-2022 2:30pm, could look as follows:

```
A  O  X  X  X  #  X  X  X  X  #  X  X  O  O
B  O  O  X  X  #  X  X  X  X  #  X  X  O  O
C  O  X  X  X  #  X  X  X  O  #  O  X  X  O
D  O  O  X  X  #  X  X  X  X  #  X  X  O  O
E  O  O  O  X  #  O  X  X  O  #  X  X  O  O
F  #  #  X  X  #  O  X  X  X  #  X  O  #  #
  01 02 03 04 05 06 07 08 09 10 11 12 13 14
```

Table 4-1

The above seating plan is stored in a file, `Monkey Goes East-202209081430.txt` as follows:

```
A,OXXX#XXXX#XXOO
B,OOXX#XXXX#XXOO
C,OXXX#XXXO#OXXO
D,OOXX#XXXX#XXOO
E,OOOX#OXXO#XXOO
F,##XX#OXXX#XO##
```

(i) Write a function `readSeatingPlan(filename)` where the parameter is a string representing the filename that the seating plan is stored. This function reads the file and store the seating plan in a dictionary. The dictionary structure after reading the file should be as follows:

```
seatingPlan = {
      'A':['O','X','X','X','#','X','X','X','X','#','X','X','O','O'],
      'B':['O','O','X','X','#','X','X','X','X','#','X','X','O','O'],
                  . . .
                  . . .

      'F':['#','#','X','X','#','O','X','X','X','#','X','O','#','#'],
            }
```

The key is the row labels, and the value is a list consisting of the seat statuses. For example, the row A consists of a list, where seat 1 is empty, 2 is booked, 3 is booked, 4 is booked and 5 is blocked etc.

(ii) Write a function `showSeatingPlan(seatingPlan)` that will display the seating plan as shown in Table 4-1. The parameter is a dictionary representation of the seating plan.

(10 marks)

(b) Write an application that manages the seat booking process for FR.

The program has a main menu, where current production titles are listed in alphabetical order, followed by the performance datetime in chronological order, as follows:

```
Main Menu - FR Productions
==========================
1. Bad Citizen @ 2022-09-08 19:30
2. Bad Citizen @ 2022-09-09 19:30
3. Monkey Goes East @ 2022-09-08 14:30
4. Monkey Goes East @ 2022-09-09 14:30
X. Exit
Enter selection: _
```

Table 4-2

Assuming user selected 1 (`Bad Citizen @ 2022-09-08 19:30`), the production title, performance datetime, seating plan and the sub-menu will be presented.

```
Production: Bad Citizen
Performance datetime: 2022-09-08 19:30
Seating plan:
A  O  O  O  O  O  #  O  X  X  X  X  #  X  O  O  O  O
B  O  O  O  O  X  #  X  X  X  O  O  #  X  O  O  O  O
C  O  O  O  O  X  #  O  X  X  O  O  #  O  O  O  O  O
D  O  O  O  O  X  #  O  O  X  X  X  #  X  O  O  O  O
  01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17

Sub-Menu
========
1. Book Seats
2. Cancel Bookings
```

```
X. Back to Main Menu
Enter option: _
```

Table 4-3

In this menu, there are options/functions that FR admin can perform: booking of seats and cancellation of bookings. The following sections will detail the setup and functionalities required for this application.

(i) Before presenting the main menu, your application needs to do the following:

- Read the file "`FR_Productions.txt`" that contains the filenames of the seating plan for all current productions:

```
Bad Citizen-202209081930.txt
Bad Citizen-202209091930.txt
Monkey Goes East-202209081430.txt
Monkey Goes East-202209091430.txt
```

- Decipher the production title and performance datetime from the filenames, such that the main menu (Table 4-2) can list all production titles in same order in the file "`FR_Productions.txt`.
- When the user selects a production, the application will proceed to load the corresponding seating plan into a dictionary, using Q4(a)(i).
- If the seating plan is successfully loaded, proceed to display the production title, performance datetime, seating plan and the sub-menu (Table 4-3).

(8 marks)

(ii) The "Book Seats" option in the sub-menu only apply to the production that is chosen at main menu.

- To book seats, user just enter the row labels and seat numbers in these formats:
  - B1 – indicating a single seat at row B, seat number 1
  - B3,B4,C3,C4 – indicating 4 seats, separated by commas

  Assume that user will enter valid input.

- This option will then validate that all seats selected must be available, i.e., not occupied or blocked before allowing this booking.
- The selected seats will be updated to X and confirmation message should be displayed.
- Below are examples of successful and unsuccessful booking:

```
Production: Bad Citizen
Performance datetime: 2022-09-08 19:30
Seating plan:
A  O  O  O  O  O  #  O  X  X  X  X  #  X  O  O  O  O
B  O  O  O  O  X  #  X  X  X  O  O  #  X  O  O  O  O
```

```
C  O  O  O  O  X  #  O  X  X  O  O  #  O  O  O  O  O
D  O  O  O  O  X  #  O  O  X  X  X  #  X  O  O  O  O
  01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17

Sub-Menu
========
1. Book Seats
2. Cancel Bookings
X. Back to Main Menu
Enter option: 1
Enter seats to book: B3,B4,A3,A4
Booking of B3,B4,A3,A4 done

Production: Bad Citizen
Performance datetime: 2022-09-08 19:30
Seating plan:
A  O  O  X  X  O  #  O  X  X  X  X  #  X  O  O  O  O
B  O  O  X  X  X  #  X  X  X  O  O  #  X  O  O  O  O
C  O  O  O  O  X  #  O  X  X  O  O  #  O  O  O  O  O
D  O  O  O  O  X  #  O  O  X  X  X  #  X  O  O  O  O
  01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17

Sub-Menu
========
1. Book Seats
2. Cancel Bookings
X. Back to Main Menu
Enter option: 1
Enter seats to book: C6,C7,C8
Exception!! C6 is BLOCKED
Exception!! C8 is BOOKED
This booking is aborted
```

- Regardless of the booking status, the application will re-display Table 4-3, pending user to select the next action/option.
- Present screenshots of successful and unsuccessful bookings.

(5 marks)

(iii) The "Cancel Bookings" option in the sub-menu apply to the production that is chosen at main menu.

- To cancel bookings, user just enter the row labels and seat numbers in these formats:
  - B1 – indicating a single seat at row B, seat number 1
  - B3,B4,C3,C4 – indicating 4 seats, separated by commas

  Assume user will enter valid input.

- This option will then validate that all seats selected must be occupied, i.e., not empty or blocked, before allowing this cancellation.
- The selected seats will be updated to O and confirmation message should be displayed.
- Below are examples of successful and unsuccessful cancellation:

```
Production: Bad Citizen
Performance datetime: 2022-09-08 19:30
Seating plan:
A  O  O  X  X  O  #  O  X  X  X  X  #  X  O  O  O  O
B  O  O  X  X  X  #  X  X  X  O  O  #  X  O  O  O  O
C  O  O  O  O  X  #  O  X  X  O  O  #  O  O  O  O  O
D  O  O  O  O  X  #  O  O  X  X  X  #  X  O  O  O  O
  01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17

Sub-Menu
========
1. Book Seats
2. Cancel Bookings
X. Back to Main Menu
Enter option: 1
Enter seats to cancel: D9,D10,D11
Cancellation of booking for D9,D10,D11 done

Production: Bad Citizen
Performance datetime: 2022-09-08 19:30
Seating plan:
A  O  O  X  X  O  #  O  X  X  X  X  #  X  O  O  O  O
B  O  O  X  X  X  #  X  X  X  O  O  #  X  O  O  O  O
C  O  O  O  O  X  #  O  X  X  O  O  #  O  O  O  O  O
D  O  O  O  O  X  #  O  O  O  O  O  #  X  O  O  O  O
  01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17

Sub-Menu
========
1. Book Seats
2. Cancel Bookings
X. Back to Main Menu
Enter option: 2
Enter seats to cancel: C7,C8
Exception!! C7 is not booked
This cancellation is aborted
```

- Regardless of the cancellation status, the application will re-display Table 4-3, pending user to select the next action/option.
- Present screenshots of successful and unsuccessful cancellations.

(5 marks)

(iv) When user exits from sub-menu, i.e., selected option “Back to Main Menu”, the application must write the latest seating plan for this production back into the file.

(2 marks)

Appendix A

The following 4 files are samples of seating plan for FR productions.

File 1: `Monkey Goes East-202209081430.txt`

```
A,OXXX#XXXX#XXOO
B,OOXX#XXXX#XXOO
C,OXXX#XXXO#OXXO
D,OOXX#XXXX#XXOO
E,OOOX#OXXO#XXOO
F,##XX#OXXX#XO##
```

File 2: `Monkey Goes East-202209091430.txt`

```
A,OOOX#XXXX#XXOO
B,OOXX#XXXO#XXOO
C,OOXX#OXXO#OOOO
D,OOXX#XXXX#XXOO
E,OOOO#OOOO#OOOO
F,##OO#OOOO#OO##
```

File 3: `Bad Citizen-202209081930.txt`

```
A,OOOOO#OXXXX#XOOOO
B,OOOOX#XXXOO#XOOOO
C,OOOOX#OXXOO#OOOOO
D,OOOOX#OOXXX#XOOOO
```

File 4: `Bad Citizen-202209091930.txt`

```
A,OOOOO#OXXXX#XOOOO
B,OOOOO#XXXOO#XOOOO
C,OOOOO#OXXOO#OOOOO
D,OOOOO#OOXXX#XOOOO
```

---- END OF ASSIGNMENT ----