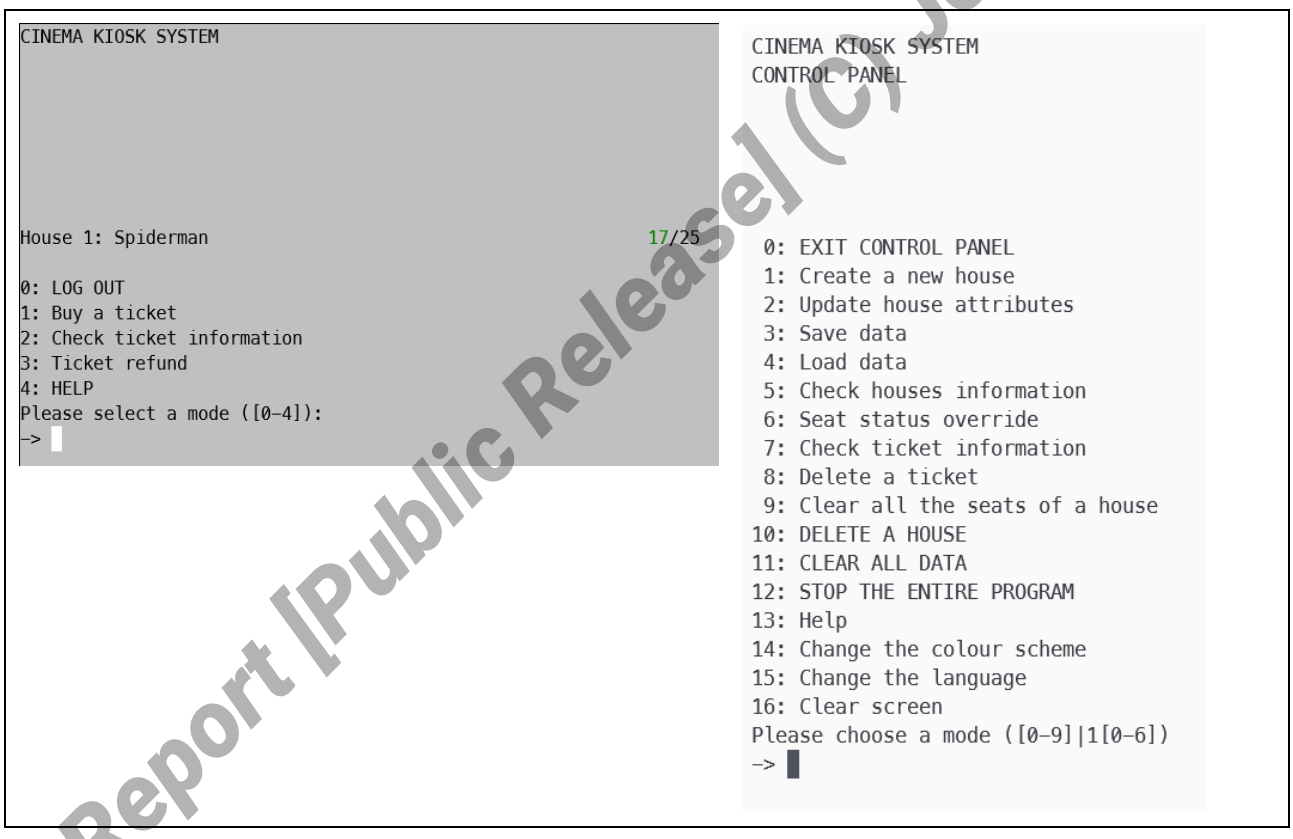
Hong Kong Diploma of Secondary Education Examination 2022-2024
Information and Communication Technology
School-based Assessment (SBA)
[redacted] School
Elective Part D - Software Development



[redacted]
Student ID: [redacted]

Contents

1 Title of SBA

Project Title: The Cinema kiosk system

The Cinema Kiosk System is a project aimed at improving the cinema experience for customers by providing a self-service kiosk system for purchasing tickets. The idea for the Cinema Kiosk System project came from the need to streamline the ticketing and concession process at cinemas. Traditionally, customers would have to wait in long lines to purchase tickets and concessions, leading to frustration and a poor customer experience. The development of self-service kiosks was seen to solve this problem and improve the overall cinema experience for customers.

The project involves the design, development, and implementation of a self-service kiosk system that allows customers to purchase tickets and concessions using a touch screen interface. The system will be integrated with the cinema's existing ticketing and point-of-sale systems, allowing for real-time updates and inventory management.

Task 1: Design and Implementation

Create a prototype of the cinema kiosk system. The prototype should include the following features:

- Login system: The kiosk should have a login system for both administrators and users. The administrator can log in with a username "admin" and password "pass," and a user can log in with a username "user" and password "bhjs."
- House and movie creation: The administrator should be able to create new houses and movies in the system. This feature will allow the administrator to add new movies and theaters to the system as needed.
- View seating plan: The kiosk should display a sold seating plan to the user. This feature will allow the user to see which seats are already taken and which are available.
- Seat selection: The user should be able to select their seat for a movie. This feature will allow the user to choose their preferred seat from the available options.
- Status storage: The kiosk should store the status of each seat, which will allow the system to keep track of which seats are sold and which are available.
- Calculation: The kiosk is able to calculate the total amount of the tickets sold in the transaction.
- You can suggest any additional features that you think would be suitable for the kiosk system to include.

Write a program to produce the competition chart.

You may want to consider some of the following key factors when designing the program:

- data structure
- variable declaration and initialization
- data collection, input and validation
- data processing
- program output
- interface of the program
- modularity
- reusability
- portability
- system development cycle
- sorting and searching algorithms

Create a presentation and/or word document to briefly describe the components involved in designing the prototype.

2 Problem definition

2.1 Define problems

Nowadays, the movie industry is rising, with more and more viewers and larger cinema. More staffs are required to handle customer services. There is often a shortage of manpower which leads to a low efficiency of ticket selling. It is difficult to find suitable employees, and it involves cost of recruiting, human resource management, training, and most importantly, wage. Also, in order to see what movies are available, customers need to queue up and ask the staff one by one. It is also very difficult for customers to know which seats are available. They have to ask one by one, which leads to an extremely poor efficiency. A long waiting time causes fewer customers willing to queue and purchase tickets. The cost of production is high, the efficiency is low. A more powerful solution is required. Hence, this movie kiosk system is born.

We need a strong and stable automated cinema kiosk system. Customers can operate the kiosk system without the assistance of staff. The kiosk system should be able to handle many tasks. This system should be able to let customers see a list of available movies, its seating plan and purchase ticket easily. Moreover, customers should be able to check the information about their purchased tickets, and be able to get refund.

This system should also provide an admin Control Panel, which allows administrator to create and modify cinema houses and do seat operations.

The advantages of using the cinema kiosk system are, customers can buy movie tickets faster. Staff (administrator) can complete complex operations more efficiently.

In admin Control Panel, there should be 16 modes:

Mode Number	Mode
0	EXIT CONTROL PANEL
1	Create a new house
2	Update house attributes
3	Save data
4	Load data
5	Check houses information
6	Seat status override
7	Check ticket information
8	Delete a ticket
9	Clear all the seats of a house
10	DELETE A HOUSE
11	CLEAR ALL DATA
12	STOP THE ENTIRE PROGRAM
13	Help
14	Change the colour scheme
15	Change the language
16	Clear screen

In user mode, there should be 4:

Mode Number	Mode
0	LOG OUT
1	Buy a ticket
2	Check ticket information
3	Ticket refund
4	HELP

2.2 Input-process-output (IPO)

When logging in:

Input	Process	Output
Username and password	Verify the user account with the data stored at `SBA/data/accounts.toml` If the username or the password is incorrect, request the user/admin to login again	(Go to the respective mode).

In user mode:

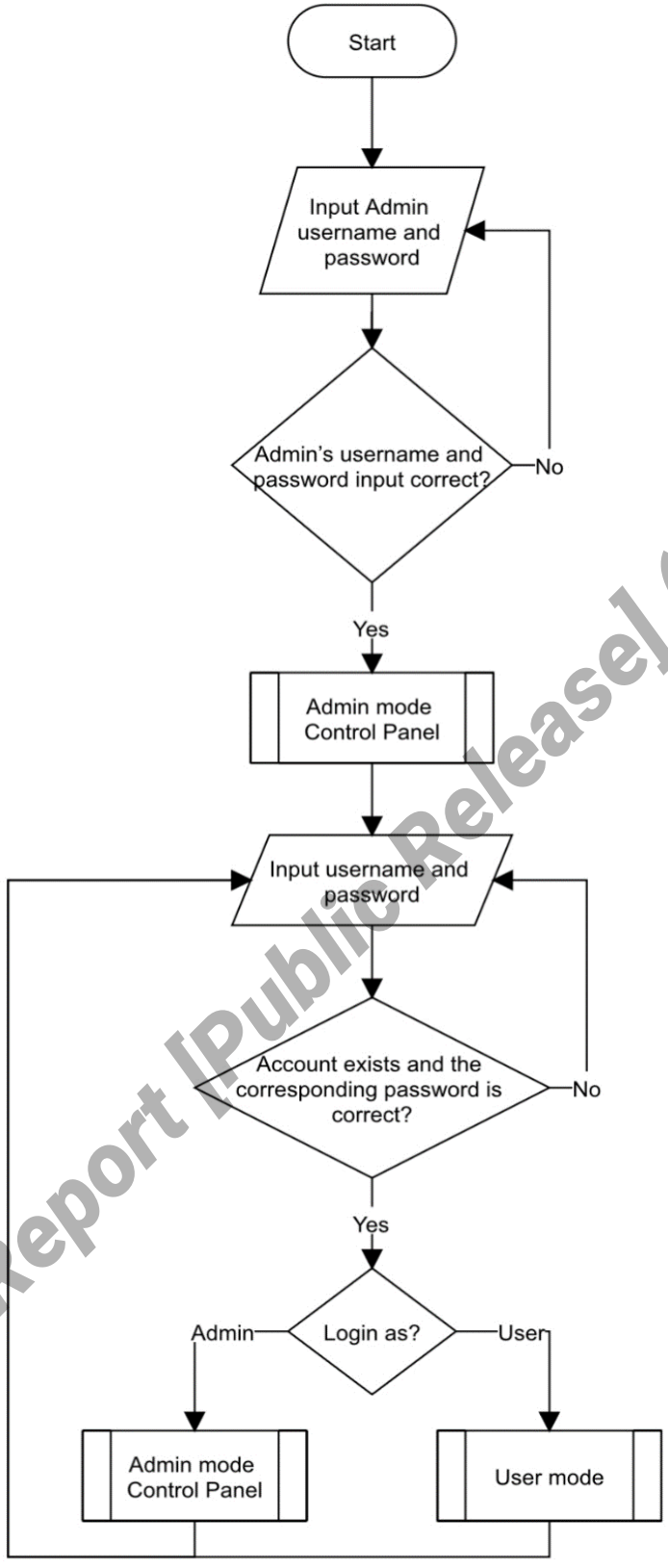
Input	Process	Output
User mode number (which operation to perform)	Match the mode number, execute the selected operation. If no such mode, admin would be asked to input the mode number again.	The message for getting the next input

In admin Control Panel:

Input	Process	Output
Admin mode number (which operation to perform)	Match the mode number, execute the selected operation. If no such mode, admin would be asked to input the mode number again.	The message for getting the next input

3 Design

3.1 Flowchart



Admin must login once at the first time, so that the administrator can initialize the system and read/write data before the user purchasing ticket. After that, it can be logged in as admin or user.

3.2 Pseudo code

Here is the initial pseudo code of the main program:

```
The main program main()
 1 | Initialize the log function
 2 | IF the Python version is older than 3.11 THEN
 3 |     exit the program
 4 | Load and initialize the language setting
 5 | Load and initialize the colour scheme
 6 | Clear the screen
 7 | Set the color of the terminal
 8 | Load saved data
 9 | WHILE TRUE DO
10 |     input username
11 |     input password
12 |     IF the username and password are correct AND it is the admin's account THEN
13 |         break the loop
14 | adminMode()
15 | WHILE TRUE DO
16 |     input username
17 |     IF the username exists THEN
18 |         input password
19 |         IF the password is correct THEN
20 |             IF the username is admin THEN
21 |                 adminMode()
22 |             ELSE
23 |                 userMode()
```

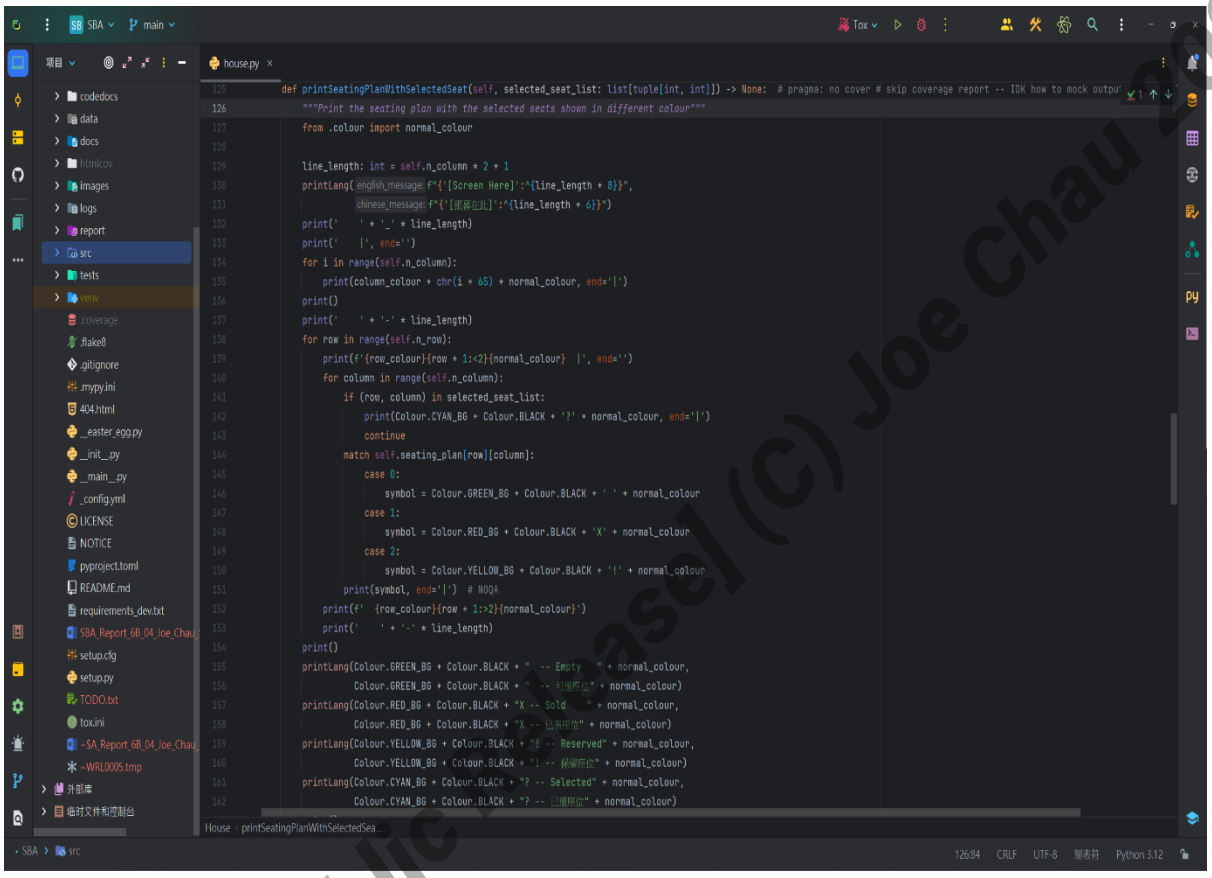
Note that the above pseudo code is not exactly the same as the code of `main()` as it was refined many times. The actual code contains error handling which is very complicated and is hard to show them in pseudocode.

4 Implementation

4.1 Integrated Development Environment (IDE)

PyCharm Professional Edition is used for developing this project.

As required, here is a screenshot of PyCharm:



PyCharm is free for its Community Edition, with the school email account, I am able to use the Professional Edition for free. PyCharm is available on Windows, macOS and Linux as an offline software.

PyCharm provides many useful functions such as code highlighting, code format checking, debugging tools, git/GitHub support, advanced inspections etc.

With the Python's built-in `breakpoint` function (and the built-in `pdb` module) and PyCharm's debug mode, debug could be done easily and unexpected behaviors could be solved quickly. Moreover, the code navigation tools and the refactoring tools are features that I frequently used. I can easily extract code to form a function. These features massively increase my development efficiency.

4.2 How to run the program

Requirements: **Python 3.11 or above.**

If *git* is installed, open *PowerShell* or *CMD* and run:

```
git clone https://github.com/Joeccp/SBA.git

python SBA
```

If not, visit *https://github.com/Joeccp/SBA/archive/refs/heads/main.zip* , the project (as a *.zip* file) should be automatically downloaded. Extract it, open the inner folder, look for the `__main__.py` file, then double click it to open it with Python.

This Python program mainly serves Windows 10 and Windows 11. But should also support Linux (roughly tested on Debian GNU/Linux 12 (bookworm) on Windows 11 x86_64 with WSL2).

4.3 Login system

The kiosk has a login system for both administrator (admin) and user. Login is required to access the system. The admin mode is called Control Panel.

User:

Username: **`user`**

Password: **`bhjs.`**

Administrator ('admin'):

Username: **`admin`**

Password: **`pass,`**

(For the avoidance of doubt, punctuations are part of the password.)

Password will not be shown when user/admin is typing (using the built-in `getpass` module). Due to the fact that this is just an SBA homework, and is in Elective Part D, usernames and hashed passwords (hashed with Secure Hash Algorithm 3,

using the built-in `hashlib.sha3_256` function) are stored locally at `SBA\data\accounts.toml` (using a `toml` file, read using the built-in `tomllib` module), instead of using any kind of web-server or database. Both admin and user cannot add or delete any account, nor change the username and password. First time logging in will require login as an administrator. Leading and trailing whitespaces are automatically ignored in the program. The program asks for input for the username and password until logged in successfully, this is basically a `DO… UNTIL…` loop.

Here are some screenshots of the login page, note that the password is already entered and not shown in the program (masked):

```
CINEMA KIOSK SYSTEM




Username: admin
Password:
```

If login as admin successfully, it will go to the admin Control Panel →

```
CINEMA KIOSK SYSTEM
CONTROL PANEL




 0: EXIT CONTROL PANEL
 1: Create a new house
 2: Update house attributes
 3: Save data
 4: Load data
 5: Check houses information
 6: Seat status override
 7: Check ticket information
 8: Delete a ticket
 9: Clear all the seats of a house
10: DELETE A HOUSE
11: CLEAR ALL DATA
12: STOP THE ENTIRE PROGRAM
13: Help
14: Change the colour scheme
15: Change the language
16: Clear screen
Please choose a mode ([0-9]|1[0-6])
->
```

Note that the 'mode hint' (`[0-9]|1[0-6]`) is a regular expression which implies that admin can enter from mode 0 to mode 16, same as the use mode below.

If login as user successfully, user mode will be activated:

```
CINEMA KIOSK SYSTEM




House 1: Spiderman                                          17/25

0: LOG OUT
1: Buy a ticket
2: Check ticket information
3: Ticket refund
4: HELP
Please select a mode ([0-4]):
->
```

If the username does not exist or the password is incorrect, admin/user has to login again:

```
CINEMA KIOSK SYSTEM


Username does not exists, please try again.


Username:
```

```
CINEMA KIOSK SYSTEM


Password incorrect, please try again.


Username:
```

The first-time logging in requires login as admin:

```
CINEMA KIOSK SYSTEM


To initialize, please login as admin.


Username:
```

If one tries to login as user at that time:

```
CINEMA KIOSK SYSTEM


Sorry, only admin can log in now as initialization is required.


Username:
```

To logout, enter mode `0` (`EXIT CONTROL PANEL` in Control Panel, `LOG OUT` in user mode), the program will go back to the login page immediately in admin Control Panel, or have to wait 3 seconds before going back to the login page as a cool down.

```
CINEMA KIOSK SYSTEM




0: LOG OUT
1: Buy a ticket
2: Check ticket information
3: Ticket refund
4: HELP
Please select a mode ([0-4]):
-> 0
You will be logged out after 3 seconds...
```

Screenshot of user logging out

Error handling in mode selection

If admin enters invalid mode number, there will be an error message and admin have to input again:

```
Please choose a mode ([0-9]|1[0-6])
-> 999
ERROR: Unknown mode code 999
```

So does the user mode:

```
CINEMA KIOSK SYSTEM



ERROR: Unknown mode



House 2: My Movie                                              25/25

0: LOG OUT
1: Buy a ticket
2: Check ticket information
3: Ticket refund
4: HELP
Please select a mode ([0-4]):
->
```

4.4 House and movie creation

A cinema 'house' is the basic component of a cinema. A `House` class was created such that each house is a unique instance of `House`.

Using classes enjoys the benefits of object-oriented programming. For example, the code has a high reusability as every instance of `House` has the same methods that have been defined only once. Also, `House` implicitly inherited from `type`, methods like `__new__` and `__eq__` can be used without defined again, as they are parts of `type`.

Here is part of the code of `House`:

```
class House:
    n_House: int = 0
    houses_table: dict[int, Self] = {}
    tickets_table: list[Ticket] = []
    total_revenue: int = 0
    total_tickets: int = 0

def __init__(self, *, row_number: int, column_number: int) -> None:
    self.n_row: int = row_number
    self.n_column: int = column_number
    self.seating_plan: Seating_plan = [[0 for _ in range(self.n_column)] for _
in range(self.n_row)]
    House.n_House += 1
    self.house_number: int = House.n_House
    House.houses_table[self.house_number] = self
    self.movie: str = ''
    self.n_seat: int = self.n_row * self.n_column
    self.house_revenue: int = 0
```

```
        self.adult_price: Price = 0
        self.child_price: Price = 0
        logger: Logger = getLogger("House.__init__")
        logger.info(f"House {self.house_number} is created --
{self.n_row}x{self.n_column}")

        # The below methods are not shown for the conciseness of this report.

        @property
        def n_available(self) -> int: ...  # Returns the number of available seats
        def clearPlan(self) -> None: ...  # Clear the seating plan
        def printSeatingPlan(self) -> None:  # Print the seating plan
        def printSeatingPlanWithSelectedSeat(self, selected_seat_list:
list[tuple[int, int]]) -> None:  # Print the seating plan with the selected seats
shown in different colour
        @classmethod
        def get_n_tickets(cls) -> int:  # Retruns the number of tickets sold in all
houses
        @classmethod
        def searchTicket(cls, target_ticket_index: int) -> Optional[Ticket]:  # Use
         binary search to search the ticket with the given ticket index
```

The administrator is able to create new houses and movies in the system. This feature will allow the administrator to add new movies and theaters to the system as needed. Every time a `Hosue` instance is created, it will register itself to `House.houses_table` with its own house index (`self.house_number`).

Each house has a 2-D rectangular seating plan (achieved by a 2-D `list` as a `House` instance's attribute: `self.seating_plan`) with 1-99 row(s) and 1-26

column(s). The size of the seating plan of a house is fixed and cannot be changed later. In terms of the 2-D list, every list inside represents a single row, with integer `0`, `1` or `2` representing Empty (i.e., available for sale), Sold (i.e., occupied and not for sale) and Reserved (i.e., reserved by administrator and not for sale) respectively.

How to create a house:

Login as administrator, enter mode `1` (`Create a new house`), enter the number of rows and columns of the house, they cannot be changed afterwards. Enter the movie name, or leave it blank (implies that the house is closed). After that, enter the price for adults and children. Note that the price can only be integers, not floats. All seats will be set as Empty by default. Here is an example screenshot:

```
CINEMA KIOSK SYSTEM
CONTROL PANEL




 0: EXIT CONTROL PANEL
 1: Create a new house
 2: Update house attributes
 3: Save data
 4: Load data
 5: Check houses information
 6: Seat status override
 7: Check ticket information
 8: Delete a ticket
 9: Clear all the seats of a house
10: DELETE A HOUSE
11: CLEAR ALL DATA
12: STOP THE ENTIRE PROGRAM
13: Help
14: Change the colour scheme
15: Change the language
16: Clear screen
Please choose a mode ([0-9]|1[0-6])
-> 1
House 2 will be the new house
Enter how many row does House 2 has (1-99): 5
Enter how many column does House 2 has (1-26): 20
Please enter the movie name (or leave it blank if the house is closed): An Amazing Movie -- 一部完美的電影
Please enter the price for adults (default is $0): $100
Please enter the price for children (default is $0): $50
Success!
```

As shown above, the movie title can include Chinese.

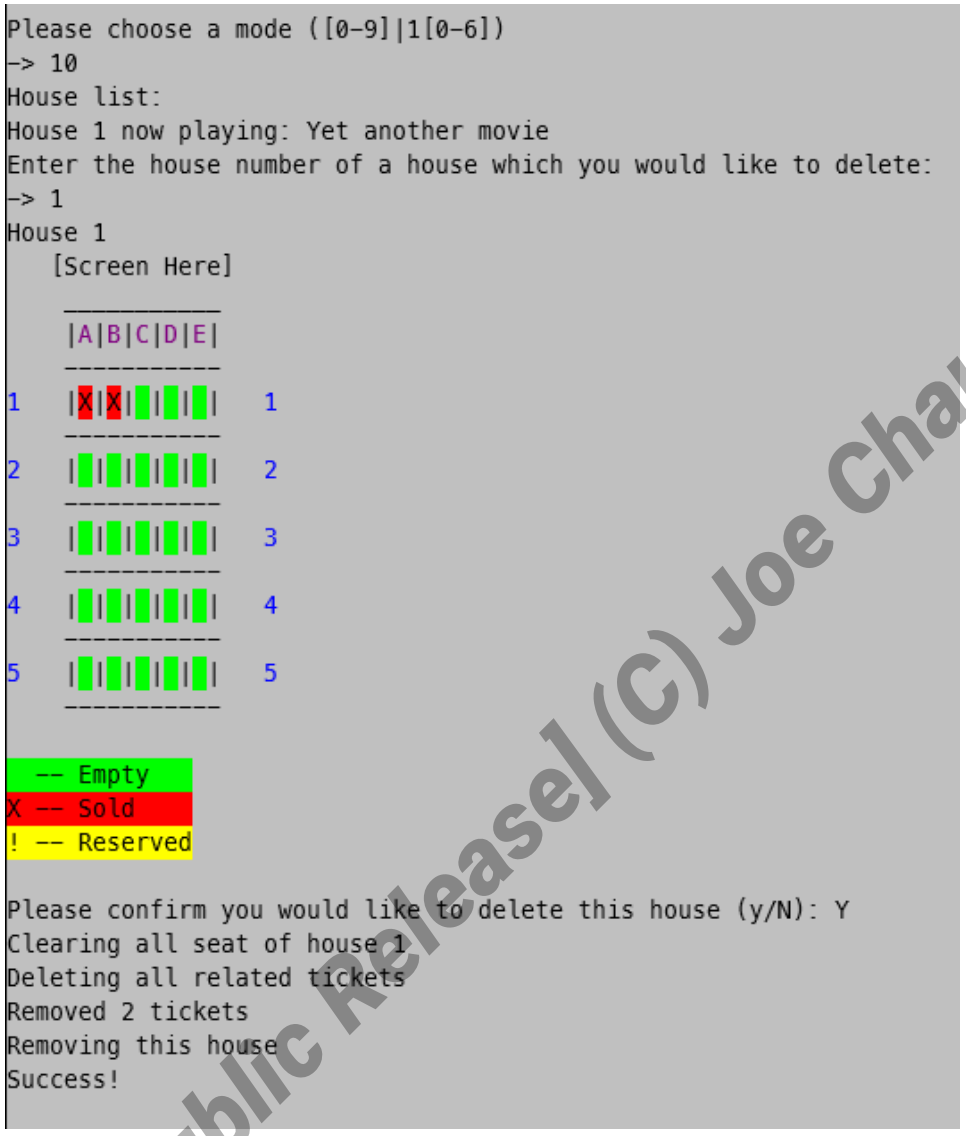
How to delete a house:

Login as an administrator, enter mode `10` (`DELETE A HOUSE`), a list of houses will be shown. Enter the house number, admin can then see the seating plan of the house. Enter 'Y' to confirm, the program will clear the seating plan of the house and delete all tickets of the house. The house number of a deleted house will not be used again, and house number of an existing house will not be changed. Note that delete a house will not affect the total revenue counter.

(Note: Not to be confused by the total revenue counter and the house revenue counter. Total revenue counter counts all revenue of all existed house. Total revenue will only change if there is a ticket purchase or refund by user.

House revenue on the other hand counts the revenue of a single house, and can be reset to zero (see the next part). Staffs(admins) can choose not to reset the house revenue counter to see the revenue of same movie of different showings inside the same house.)

An example screenshot of deleting a house:



How to change or update the information (attributes) of a house:

Login as administrator, enter mode `2` (`Update house attributes`), admin can see a list of houses, enter the house number, enter the new movie name (or leave it blank to implies no movie will be played, i.e., the house is closed). Enter 'Y' if admin wants to update the price (or 'N' if not), enter the price for adults and children respectively. Enter 'Y' if admin wants to reset the house revenue counter to zero (or 'N' if not). Enter 'Y' if admin wants to clear all seats (i.e., seats in the seating plan) and tickets from that house (or 'N' if not).

Here is a screenshot of an example of admin updating House 2: (Note that the 'old movie name' shown below is '`(None)`', implies that House 2 was originally closed.)

```
Please choose a mode ([0-9]|1[0-6])
-> 2
House list:
House 2 is closed
Please select the house:
-> 2
Please enter the movie name (or leave it blank if no movie will be played): My Movie
Successfully changed the movie in house 2!
(None) --> My Movie
Would you like to update the price too?(y/N)y
Please enter the price for adults (default not to change as $199): $200
Please enter the price for children (default not to change as $99): $100
Do you want to reset the house revenue counter to zero? (y/N)Y
Would you like to clear all seats and tickets from this house too? (y/N)Y
Clearing all seat of house 2
Success!
Deleting all related tickets
Removed 0 tickets
```

4.5 View seating plan

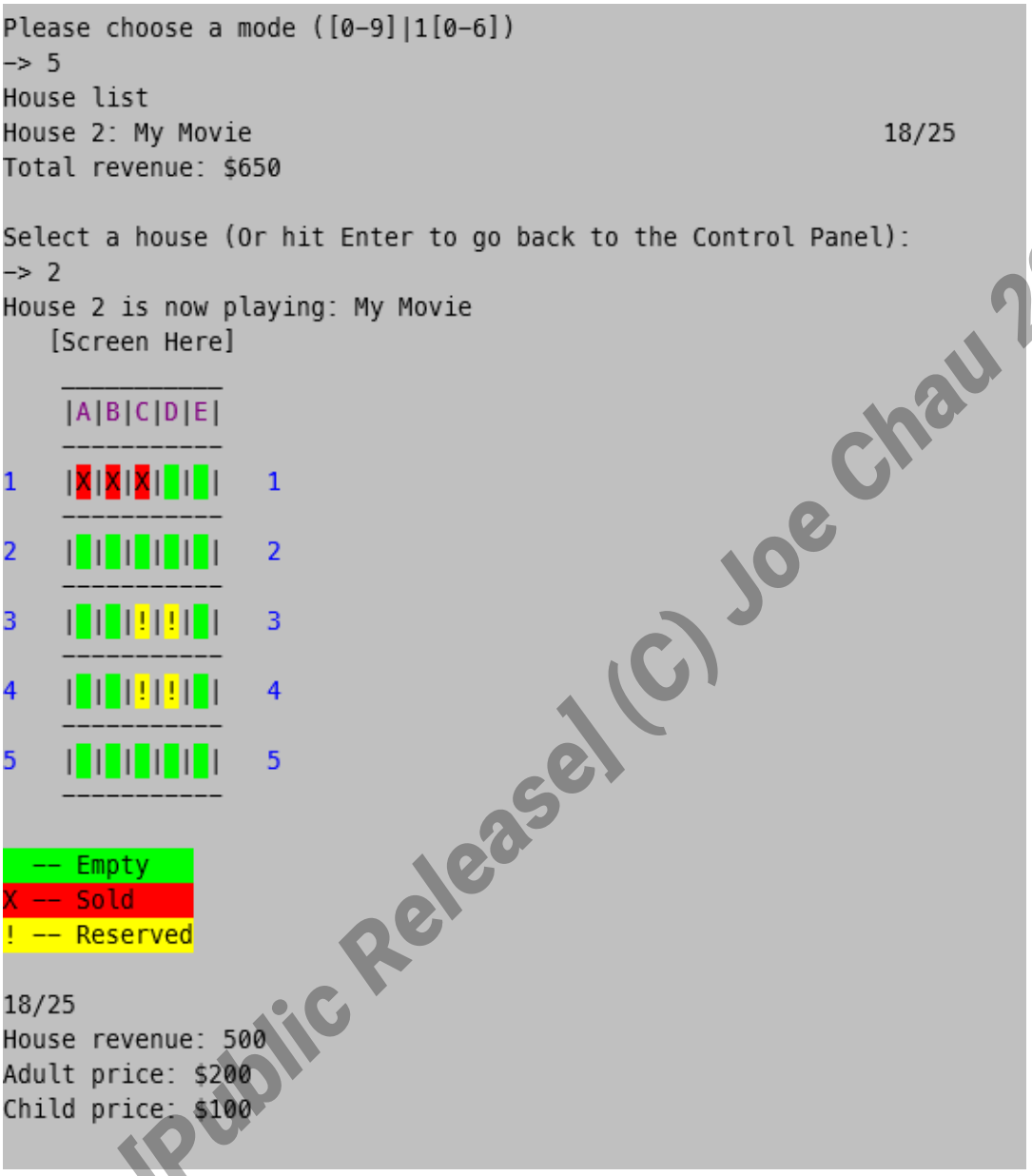
The kiosk is able to display a sold seating plan to the user. This feature will allow the admin and user to see which seats are already taken and which are available.

In the seating plan, horizontal rows are ordered top-down sequentially with numbers starting from 1 (*row number*), vertical columns are numbered from left to right with alphabets starting from 'A'. The screen is located on top of the seating plan labelled with `[Screen Here]`, i.e., the screen is near the first row. Green cells are Empty seats, red cells marked with a cross ('`X`') are Sold seats, yellow cells mark with an exclamation mark ('`!`') are Reserved seats.

How to see the seating plan as an administrator

Login as an administrator, enter mode `5` (`Check houses information`), a list of houses should be printed, enter the house number, then the seating plan will be shown.

Here is an example screenshot:



How to see the seating plan as a user

Login as a user, user can see a list of available houses and houses that are full. (‘*available house*’ refers to house that has at least 1 empty seat, and has a movie playing (i.e., has a movie name).) Enter mode `1` (`Buy a ticket`), user can see a list of available houses again. Enter the house number, if user just want to see the seating plan and doesn’t want to buy any tickets, enter 0 for the number of adult and child ticket.

Here are some example screenshots to see the seating plan of House 2, note that House 1 was deleted, House 2 is available, House 3 is closed and House 4 is full:

```
House 2: My Movie                                          25/25
House 4: An Amazing Movie                                  0/25

0: LOG OUT
1: Buy a ticket
2: Check ticket information
3: Ticket refund
4: HELP
Please select a mode ([0-4]):
-> 1
```

```
House(s) available:
House 2: My Movie                                          25/25

Please enter the house number (or hit Enter to go back to the main menu):
-> 2
```

4.6 Clear the seating plan

Administrator is able to clear the seating plan. This feature will allow the administrator to update the house if new showing of the same movie and house will be played.

How to clear the seating plan (i.e., empty all the seats) of a house:

Login as an administrator, enter mode `9` (`Clear all the seats of a house`), admin should be able to see a list of houses, enter the house number, admin should see the seating plan of the house. Confirm by entering 'Y', (or 'N' if not) to clear the seating plan.

Here is an example screenshot:

```
Please choose a mode ([0-9]|1[0-6])
-> 9
House list:
House 2 now playing: My Movie
House 3 is closed
House 4 now playing: An Amazing Movie
Enter the house number of a house which you would like to empty:
-> 4
House 4
   [Screen Here]
    ___________
    |A|B|C|D|E|
    -----------
1   |X|X|X|X|X|   1
    -----------
2   |X|X|X|X|X|   2
    -----------
3   |X|X|X|X|X|   3
    -----------
4   |X|X|X|X|X|   4
    -----------
5   |X|X|X|X|X|   5
    -----------

  -- Empty
X -- Sold
! -- Reserved

Please confirm you would like to clear all seats and tickets of this house (y/N): Y
Clearing all seat of house 4
Deleting all related tickets
Removed 0 ticket
```

Note that the program said that it 'removed 0 tickets' is because I overrode all the seats in House 4 with admin mode `6` (more on that below) in order to demonstrate.

4.7 Coordinate expression

Coordinate expressions are used to represent a range of seats.

A coordinate expression can be:

1. A single seat coordinate with row number and column number.

   E.g. `1A`, `2B`, `P35`, `35P`, `99Z`

2. Two seat coordinates separated by a colon ('`:`'), represents a rectangular range of seats. The first seat coordinate is the coordinate of the top-left

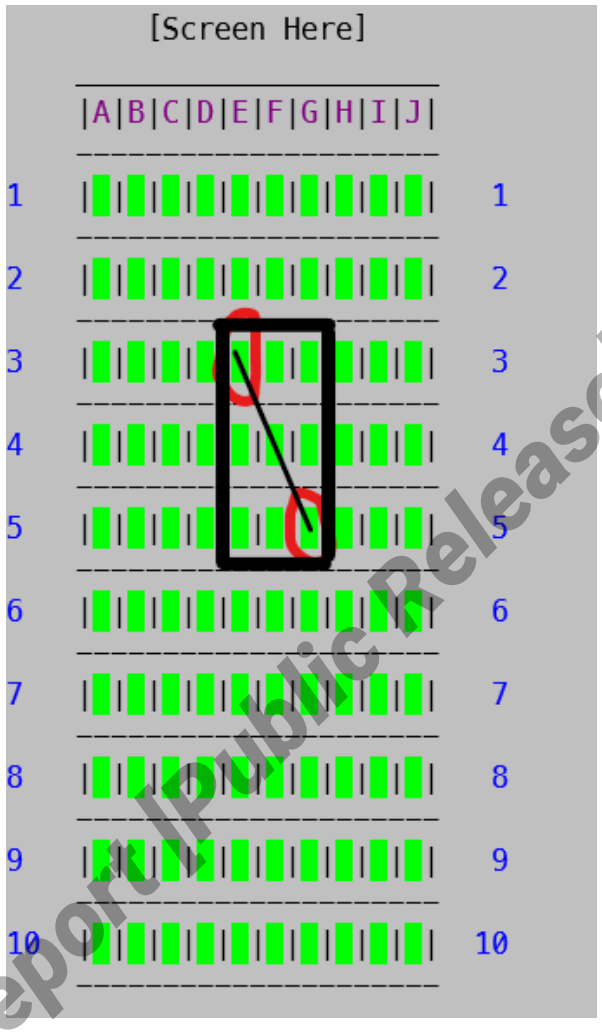
corner and the second seat coordinate is the coordinate of the bottom-right corner of the area:

i. These two coordinates must not be the same seat.

ii. The two coordinates must be two complete coordinates. (i.e. no missing row / column number in these two coordinates)

iii. Valid examples: `1A:99Z`, `1B:2A`, `1A:1B`, `1A:2A`, `P35:P45`, `35P:45P`.

For example, coordinate expression `3E:5G` represents the following area:



`3E` marks the top-left seat of the area, while `5G` marks the bottom-right seat of the area. Together, they form a diagonal of a rectangular area.

(Just to be clear, coordinate expression can represent seats in a single row or column. For example, coordinate expression `1A:1B` marks the seats `1A` and `1B`.)

To get the a list of seats that a coordinate expression represents, it will be passed to `getCoorsFromCoorExpr()` along with the number of rows and columns of the house as arguments. It calls `coorExprAnalysis()` to check whether the coordinate expression is valid and get the corner coordinates. If invalid, it raises exceptions that are inherited from `CoordinateExpressionException`:

Exception Name	**Meaning**
`EmptyCoordinate`	Empty coordinate
`InvalidCharacter`	Invalid character inside the coordinate expression
`MoreThanOneColon`	More than one colon in a coordinate expression
`NoColumnCoordinate`	No column coordinate
`NoRowCoordinate`	No row coordinate
`AlphabetCharacterInRowNumber`	Column coordinate has more than two characters
`RowNumberIsZero`	Row number is 0
`NoStartingCoordinate`	Starting coordinate not given
`NoEndingCoordinate`	Ending coordinate not given
`CoordinatesWrongOrder`	Wrong order of the two coordinates
`SameCoordinate`	Two same coordinates
`RowNumberOutOfRange`	The row number is out of range, and does not exist
`ColumnNumberOutOfRange`	The column number is out of range, and does not exist
`RowCoordiantesAtTwoSide`	Two row coordinates given in a single seat coordinate
`ColumnCoordinatesAtTwoSide`	Two column coordinates given in a single seat coordinate

Errors are represented by exceptions, instead of special return values, so that the program can make use of the `else` and `finally` clause of the `try/except` block to do clean-up actions.

Note that although `getCoorsFromCoorExpr()` and `coorExprAnalysis()` does not use regular expression to handle the coordinate expression. A regular expression of coordinate expression for reference would be:

```
/^([1-9][0-9]?[A-Z]|[A-Z][1-9][0-9]?|[1-9][0-9]?[A-Z]:[1-9][0-9]?[A-Z]|[1-9][0-
9]?[A-Z]:[A-Z][1-9][0-9]?|[1-9][0-9]?[A-Z]:[1-9][0-9]?[A-Z]|[1-9][0-9]?[A-Z]:[1-
9][0-9]?[A-Z])$/i
```

where **all spaces are ignored** and deleted before matching.

4.8 Seat selection

The user should be able to select their seat for a movie. This feature will allow the user to choose their preferred seat from the available options.

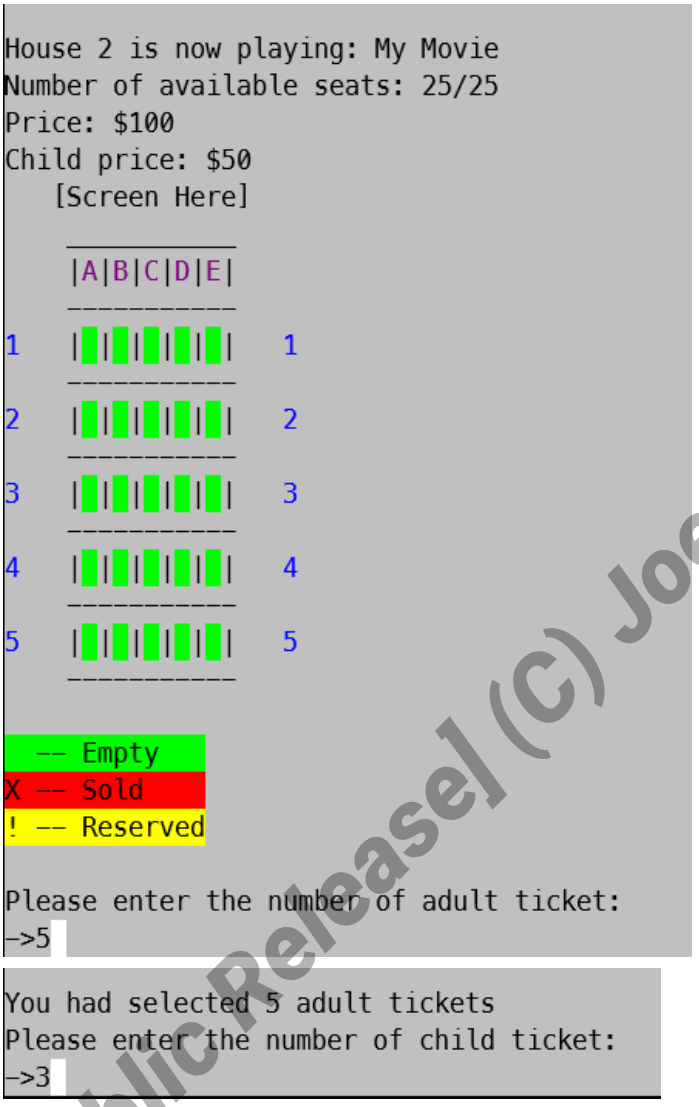
How to buy a seat as user:

Login as a user, enter mode `1` (`Buy a ticket`), user can see a list of available houses and houses that are full. Enter the house number, user can see a list of available houses again.

```
Please select a mode ([0-4]):
-> 1
```

```
House(s) available:
House 2: My Movie                                          25/25

Please enter the house number (or hit Enter to go back to the main menu):
-> 2
```

Enter the number of adult ticket(s) and the number of child ticket(s):

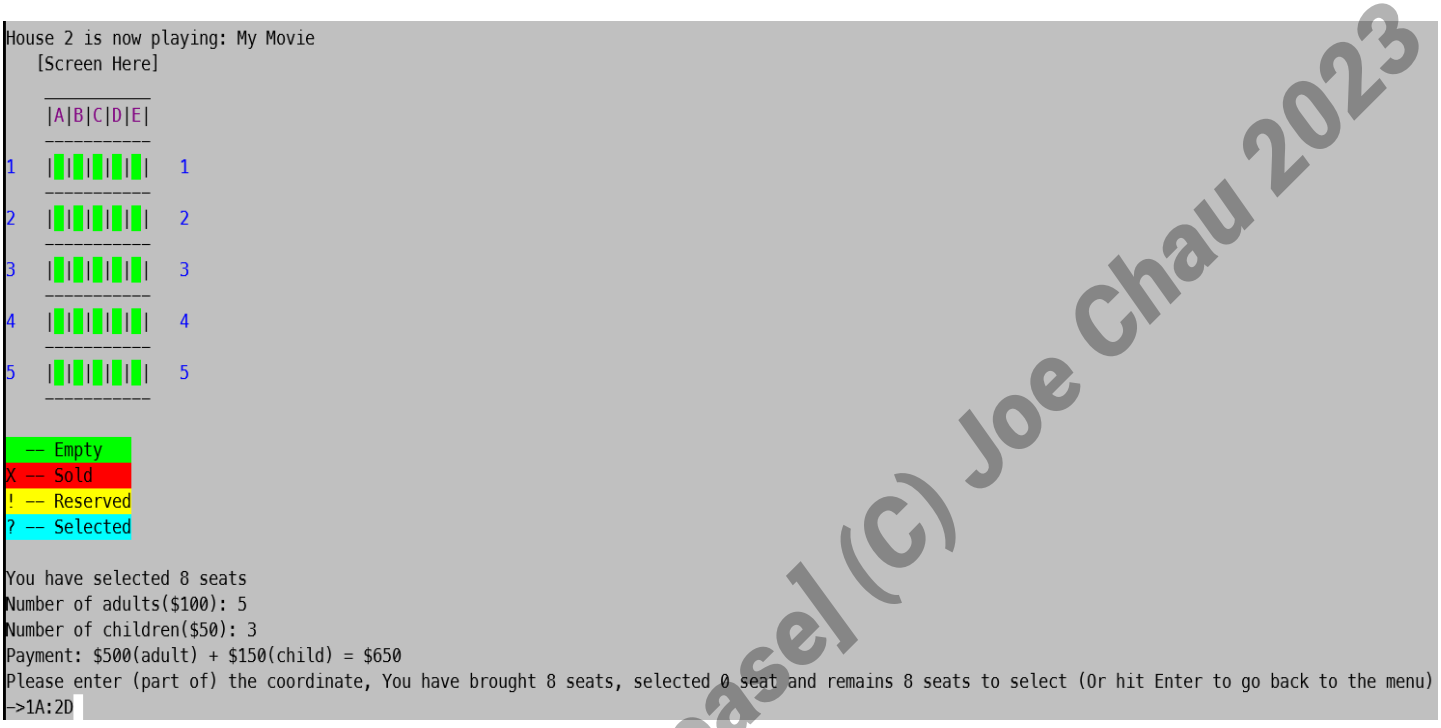


The total price will be shown as follow as follow, confirm by entering 'Y' (or 'N' if not):

```
You have selected 8 seats
Number of adults($100): 5
Number of children($50): 3
Payment: $500(adult) + $150(child) = $650
Please confirm: [Y/n]
```

Enter the coordinate expression to select seats, user can enter more than 1 time, user can split an area of seats to 2 or more coordinate expressions, so that user

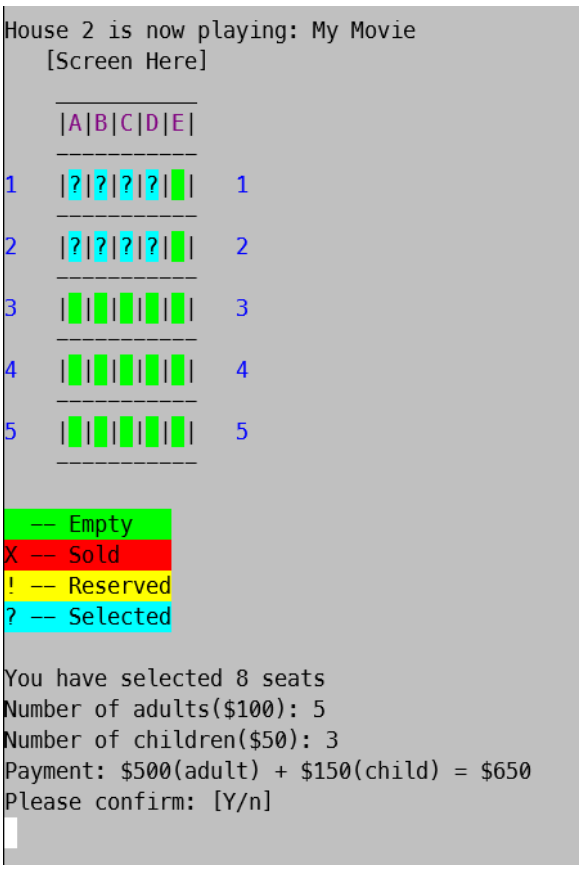
can purchase seats that does not form a rectangular area. User cannot enter coordinate expression that contains Sold seats nor Reserved Seats. If user selected seats that are already selected, they will remain selected. User cannot deselect a selected seat.



In this example, the user entered `1A:2D`.

The total number of selected seats matches the number of tickets entered.

The program will ask for confirmation once again. Enter 'Y' to confirm or 'N' to cancel the purchase.

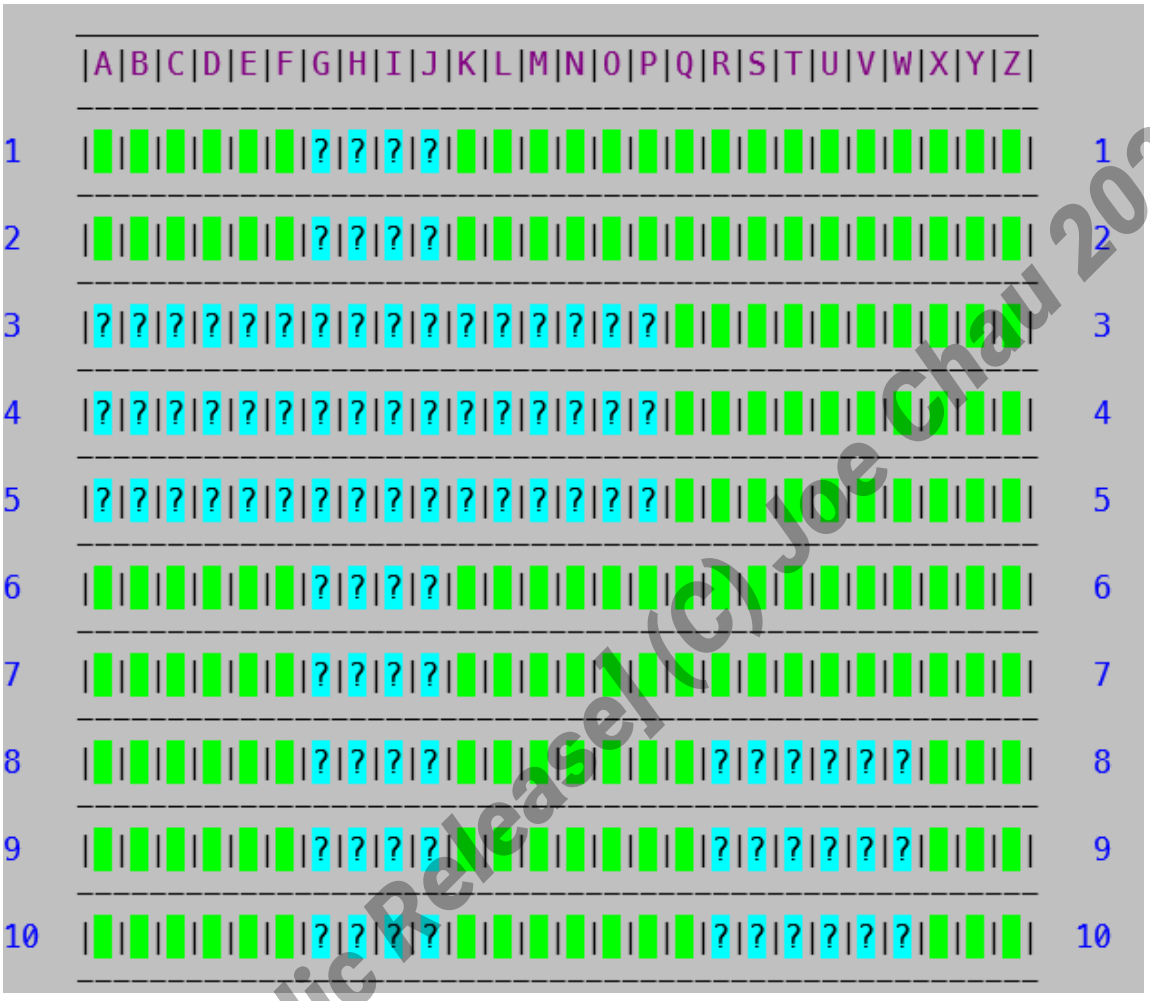


After that, the purchase is successful, the program will then print out all tickets (more on tickets below). Notice that the if there are both adults and children, the seats in the front near the screen will be assigned to the children.

```
Your ticket:
T00006 @2023-11-21T20:21:44: House 2  -- My Movie                ~Seat<1A> $50

T00007 @2023-11-21T20:21:44: House 2  -- My Movie                ~Seat<1B> $50

T00008 @2023-11-21T20:21:44: House 2  -- My Movie                ~Seat<1C> $50

T00009 @2023-11-21T20:21:44: House 2  -- My Movie                ~Seat<1D> $100

T00010 @2023-11-21T20:21:44: House 2  -- My Movie                ~Seat<2A> $100

T00011 @2023-11-21T20:21:44: House 2  -- My Movie                ~Seat<2B> $100

T00012 @2023-11-21T20:21:44: House 2  -- My Movie                ~Seat<2C> $100

T00013 @2023-11-21T20:21:44: House 2  -- My Movie                ~Seat<2D> $100



Thank you for your purchase!

Hit Enter to go back to the main menu
```

If the user enters something invalid (e.g., violates the above rules of selecting seats), the purchase will be cancelled, and user will go back to the user mode menu with the corresponding error message shown. However, if the error is a `CoordinateExpressionException`, user is allowed to enter the coordinates expression again with the error message shown above the seating plan.

For example, the below area of seats can be selected by entering `3A:5P`, `1G:10J` and `8R:10W`.



4.9 Admin seat override

The admin can override the seat status of any seat. Seats brought / reserved/ emptied by admin seat override operation will not generate, modify nor delete any ticket. Staff (admin) should check the status of the seat manually before changing the status of a seat.

How to override seat status

To do so, login as administrator, enter mode `6` (`Seat status override`), enter the command with the following format:

```
(EMPTY | BUY | RESERVE) - <House number> - <Coordinate Expression>
```

Replace `(EMPTY | BUY | RESERVE)` to the action wanted to be performed (`EMPTY` / `BUY` / `RESERVE`);

replace `<House number>` to the house number;

replace `<Coordinate Expression>` to the coordinate expression of the seat(s) admin wants to override.

Three arguments must be separated with hyphens ('–'), as shown above.

The command is case-insensitive. All spaces are ignored.

Admin should not include brackets ('`()`', '`<>`') and vertical bar ('`|`'). They are purely for demonstration purpose only, though they will be ignored.

For example, suppose all seats in House 2 were empty originally, as below:

```
Please choose a mode ([0-9]|1[0-6])
-> 5
House list
House 2: My Movie                                              25/25
Total revenue: $650

Select a house (Or hit Enter to go back to the Control Panel):
-> 2
House 2 is now playing: My Movie
   [Screen Here]

    ___________
    |A|B|C|D|E|
    -----------
1   |_|_|_|_|_|   1
    -----------
2   |_|_|_|_|_|   2
    -----------
3   |_|_|_|_|_|   3
    -----------
4   |_|_|_|_|_|   4
    -----------
5   |_|_|_|_|_|   5
    -----------

  -- Empty
X -- Sold
! -- Reserved

25/25
House revenue: 0
Adult price: $100
Child price: $50
```

If the administrator wants to reserve all seats in House 2, he should enter the following:

```
Please choose a mode ([0-9]|1[0-6])
-> 6
Note: Seats brought / reserved / emptied from this control panel DO NOT have / WILL NOT delete a ticket.
Staffs should check the status of the seat manually before changing the status of a seat.
The mode will NOT check the seat status for you.
Command format:

(EMPTY | BUY | RESERVE) - <House number> - <Coordinate Expression>
(or hit Enter to go back to the Control Panel menu)
-> reserve -2 -1A:5E
Success!

25 seats overwritten.
```

Where `2` is the house number and `1A:5E` is the coordinate expression. A “Success!” message will be shown if the operation success. Admin can confirm with admin mode `5`.

```
Please choose a mode ([0-9]|1[0-6])
-> 5
House list
House 2: My Movie                                          0/25
Total revenue: $650

Select a house (Or hit Enter to go back to the Control Panel):
-> 2
House 2 is now playing: My Movie
   [Screen Here]

    ___________
    |A|B|C|D|E|
    -----------
1   |!|!|!|!|!|   1
    -----------
2   |!|!|!|!|!|   2
    -----------
3   |!|!|!|!|!|   3
    -----------
4   |!|!|!|!|!|   4
    -----------
5   |!|!|!|!|!|   5
    -----------

  -- Empty
X -- Sold
! -- Reserved

0/25
House revenue: 0
Adult price: $100
Child price: $50
```

Notice that after the operation, the total revenue remains unchanged. The house revenue should also remains unchanged, even if the admin override all the seat status to sold:

```
Please choose a mode ([0-9]|1[0-6])
-> 6
Note: Seats brought / reserved / emptied from this control panel DO NOT have / WILL NOT delete a ticket.
Staffs should check the status of the seat manually before changing the status of a seat.
The mode will NOT check the seat status for you.
Command format:

(EMPTY | BUY | RESERVE) - <House number> - <Coordinate Expression>
(or hit Enter to go back to the Control Panel menu)
-> buy -2 -1A:5E
Success!

25 seats overwritten.
```

```
Please choose a mode ([0-9]|1[0-6])
-> 5
House list
House 2: My Movie                                          0/25
Total revenue: $650

Select a house (Or hit Enter to go back to the Control Panel):
-> 2
House 2 is now playing: My Movie
   [Screen Here]

    ___________
    |A|B|C|D|E|
    -----------
1   |X|X|X|X|X|   1
    -----------
2   |X|X|X|X|X|   2
    -----------
3   |X|X|X|X|X|   3
    -----------
4   |X|X|X|X|X|   4
    -----------
5   |X|X|X|X|X|   5
    -----------

  -- Empty
X -- Sold
! -- Reserved

0/25
House revenue: 0
Adult price: $100
Child price: $50
```

4.10 Tickets

Tickets are generated when a user has bought a seat, each ticket represent a single seat and indicates the following information about a purchase of a movie ticket, which includes:

- The date and time of the purchase
- House number of the movie
- Movie name
- Seat number (The row and column number of the seat)
- Price
- Ticket number
- Ticket index (ticket number as a number, without the 'T' prefix)

Each ticket has a unique ticket number which starts with a 'T', followed by at least 5 digits, starts from `T00001`.

The type hint of a ticket (`Ticket`) is:

```
Ticket_index: TypeAlias = int

Ticket_number: TypeAlias = str

Time: TypeAlias = str

House_number: TypeAlias = int

Movie: TypeAlias = str

Row_number: TypeAlias = int

Column_number: TypeAlias = int

Price: TypeAlias = int

Ticket: TypeAlias = tuple[Ticket_index, Ticket_number, Time, House_number,

Movie, Row_number, Column_number, Price]
```

All tickets (as `tuple`) are stored in a `list` (`House.tickets_table`) sorted by the ticket number.

How the program searches for a ticket by its ticket number

The program uses binary search (with `House.searchTicket()`, a `classmethod` of `House`) to search for a ticket, or to confirm whether it exists or not, by its ticket number. Here is code of `House.searchTicket()`:

```
    @classmethod
    def searchTicket(cls, target_ticket_index: int) -> Optional[Ticket]:
        """
        Searches the ticket with the given ticket index, and returns it.
        Assumes the (format of the) ticket index is valid.
        If the ticket does not exist, returns None.

        It uses binary search.

        ASSUMES House.tickets_table IS ALREADY SORTED (As it should be sorted
anytime).

        :param target_ticket_index: Ticket index with a valid format
        :type target_ticket_index: int
        :return: Ticket
        :rtype: Optional[tuple[int, str, str, int, str, int, int, int]]
        """
        logger: Logger = getLogger("House.searchTicket")
        logger.info(f"Searching ticket: {target_ticket_index}")

        if not cls.tickets_table:
            logger.info("No any tickets, just return None")
            return None
```

```
        min_: int = 0
        max_: int = cls.get_n_tickets() - 1
        logger.debug(f"Min: {min_}  & Max: {max_}")

        while True:
            logger.debug(f"Min: {min_}  & Max: {max_}")
            if min_ == max_:
                if cls.tickets_table[min_][0] == target_ticket_index:
                    logger.debug(f"cls.tickets_table[min_][0] =
{cls.tickets_table[min_][0]}")
                    logger.info(f"Return ticket: {cls.tickets_table[min_]}")
                    return cls.tickets_table[min_]
                else:
                    logger.info("No such ticket")
                    return None

            if max_ - min_ == 1:
                if cls.tickets_table[max_][0] == target_ticket_index:
                    return cls.tickets_table[max_]
                elif cls.tickets_table[min_][0] == target_ticket_index:
                    return cls.tickets_table[min_]
                else:
                    return None

            half: int = int((max_ + min_) / 2)  # int() = floor()
            guess_ticket_index: int = cls.tickets_table[half][0]

            logger.debug(f"half: {half}  guess_ticket_index:
{guess_ticket_index}")
```

```
                if guess_ticket_index == target_ticket_index:
                    logger.info(f"Return ticket: cls.tickets_table[half] =
{cls.tickets_table[half]}")
                    return cls.tickets_table[half]
                elif guess_ticket_index > target_ticket_index:
                    max_: int = half
                    logger.debug("max_ <- half")
                    logger.debug(f"max is now {max_}")
                else:
                    min_: int = half
                    logger.debug("min_ <- half")
                    logger.debug(f"min is now {min_}")
```

How to check ticket's information as a user

Login as a user, enter mode `2` (`Check ticket information`).

```
Please select a mode ([0-4]):
-> 2
```

Enter the ticket number, if the ticket exists, its information will be shown:

```
CINEMA KIOSK SYSTEM




Please enter your ticket number (starts with 'T'):
-> T00006
T00006 @2023-11-21T20:21:44 House 2  -- My Movie                    ~ Seat<1A> $50



Hit Enter to go back to the main menu
```

If not, a “No such ticket” message will be shown:

```
CINEMA KIOSK SYSTEM




Please enter your ticket number (starts with 'T'):
-> T99999

No such ticket



Hit enter to go back to the main menu
```

How to get refund of a ticket as a user

Login as a user, enter mode `3` (`Ticket refund`), enter the ticket number.

Information of the ticket will be printed on the screen. Confirm by entering ‘Y’ (or ‘N’ if not).

```
Please select a mode ([0-4]):
-> 3
```

```
Please enter your ticket number (starts with 'T'):
-> T00006

T00006 @2023-11-21T20:21:44 House 2  -- My Movie                    ~ Seat<1A> $50

Are you sure you want to get refund of this ticket? (y/N)
->
```

If the ticket does not exist, a “No such ticket” message will be shown.

User get refund of a ticket will affect the house revenue and the total revenue.

How to check ticket's information as an admin

Administrator can choose to see all active tickets' (i.e., tickets that are not refunded by user or deleted by administrator) information or see the information of a specific ticket. Administrator can also see the number of active tickets and the total number of tickets sold historically.

Login as an administrator, enter mode `7` (`Check ticket information`), hit enter to see all active ticket's information;

```
Please choose a mode ([0-9]|1[0-6])
-> 7
Enter the ticket number to see the information about that ticket,
or hit enter to see all ticket information
->
T00004 @2023-11-21T15:37:52 House 2  -- My Movie                ~Seat<1B> $200
T00007 @2023-11-21T20:21:44 House 2  -- My Movie                ~Seat<1B> $50
T00008 @2023-11-21T20:21:44 House 2  -- My Movie                ~Seat<1C> $50
T00009 @2023-11-21T20:21:44 House 2  -- My Movie                ~Seat<1D> $100
T00010 @2023-11-21T20:21:44 House 2  -- My Movie                ~Seat<2A> $100
T00011 @2023-11-21T20:21:44 House 2  -- My Movie                ~Seat<2B> $100
T00012 @2023-11-21T20:21:44 House 2  -- My Movie                ~Seat<2C> $100
T00013 @2023-11-21T20:21:44 House 2  -- My Movie                ~Seat<2D> $100
TOTAL: 8 tickets active, 13 tickets were created.
```

or enter the ticket number to search for a specific ticket.

```
Please choose a mode ([0-9]|1[0-6])
-> 7
Enter the ticket number to see the information about that ticket,
or hit enter to see all ticket information
-> T00007
T00007 @2023-11-21T20:21:44 House 2  -- My Movie                ~Seat<1B> $50
TOTAL: 8 tickets active, 13 tickets were created.
```

A "No such ticket" message will be shown if the target ticket does not exist.

How to delete a ticket as an administrator

Login as an administrator, enter mode `8` (`Delete a ticket`), enter the ticket number

```
Please choose a mode ([0-9]|1[0-6])
-> 8
Please enter the ticket number (starts with 'T'):
-> T00008
T00008 @2023-11-21T20:21:44 House 2  -- My Movie                    ~Seat<1C> $50
Successfully deleted this ticket
```

A “No such ticket” message will be shown if the target ticket does not exist. House revenue and total revenue will not be affected.

Errors

In the above four different kinds of ticket operations, if the format of the ticket number is invalid, user/admin will go back to the menu. An error message will be shown. Here is a list of possible errors:

Example input	Error message
T00000	ERROR: Invalid ticket number -- ticket number is all zero
abcdefg	ERROR: Invalid ticket number format -- ticket number starts with 'T'
T0001	ERROR: Invalid ticket number -- ticket number too short
T00000001	ERROR: Invalid ticket number -- more than 4 leading zeros
TABCDE	ERROR: Invalid ticket number -- ticket number should be a single character 'T' followed by decimal numbers
T	ERROR: Invalid ticket number -- ticket number has no decimal numbers

4.11 Status storage

The kiosk is able to store the status of each seat, which will allow the system to keep track of which seats are sold and which are available. All data are automatically saved. Each time saving data will cover the previous saved data.

Houses and tickets data are stored at `SBA/data/houses` and `SBA/data/tickets` (without a filename extension), the colour scheme setting is stored at `SBA/data/colour.txt`, the language option is stored at `SBA/data/language.txt` if there are no such files, the program will create them if needed.

To manually save data, login as administrator, enter mode `3` (`Save data`).

```
Please choose a mode ([0-9]|1[0-6])
-> 3
Reaching the data folder
Writing houses data
Writing tickets data
Writing colour scheme setting
Writing language option
Data saving process finished
```

To manually load data, login as administrator, enter mode `4` (`Load data`).

```
Please choose a mode ([0-9]|1[0-6])
-> 4
Finding houses data
Houses data loaded
Finding tickets data
Tickets data loaded
Loading colour scheme
Colour scheme loaded
Loading language setting
Colour scheme loaded
Data loading process finished
```

If there are no such data, the program will create them with the current data.

```
Please choose a mode ([0-9]|1[0-6])
-> 4
Finding houses data
No houses data found
Finding tickets data
No tickets data found
Loading colour scheme
Colour scheme loaded
Loading language setting
Colour scheme loaded
Data loading process finished
```

Technical details

`[House.total_revenue, House.houses_table, House.n_House]` is pickled and dumped into `SBA/data/houses` with the built-in `pickle.dump` function. `[House.total_tickets, House.tickets_table]` is pickled and dumped into `SBA/data/tickets` also with the built-in `pickle.dump` function. `SBA/data/houses` and `SBA/data/tickets` are binary files, containing pickled Python objects. Those files are NOT human-readable. To read these file, the program uses the built-in `pickle.load` function.

For reference, the official documentation about pickling is available at https://docs.python.org/3/library/pickle.html.

How to clear all data

The administrator is able to clear all data, i.e., to reset all options.

Login as administrator, enter mode `11` (`CLEAR ALL DATA`), enter 'Y' to confirm (or 'N' to cancel).

```
Please choose a mode ([0-9]|1[0-6])
-> 11
Please confirm you would like to clear ALL data (y/N): Y
Successfully removed unsaved tickets data
No saved tickets data
Process of removing tickets data finished
Successfully unsaved local houses data
No saved houses data
Process of removing tickets houses finished
Resetting the colour scheme to DARK
Successfully deleted colour scheme setting file
Process of removing colour scheme setting file finished finished
Resetting the language to ENGLISH
Process of removing language setting file finished finished
Finish!
```

As shown above, clearing all data will remove all tickets data, houses data and reset the colour scheme and language option.

4.12 Language

This program supports traditional Chinese. However, only administrator has the permission to change the language at this moment (as all accounts use the same setting file).

Changing the language in admin Control Panel will also affects the language of user mode and login page.

To change the language, login as administrator, enter mode `15` (`Change the language`, or `轉換語言` in traditional Chinese).

Python uses UTF-8 as its default encoding, so traditional Chinese and English should be printed properly.

The Control Panel in traditional Chinese:

```
Please choose a mode ([0-9]|1[0-6])
-> 15
成功！ 現在的語言為中文


 0：退出控制面板
 1：創建新電影院
 2：更新電影院屬性
 3：儲存資料
 4：載入資料
 5：查詢電影院資訊
 6：覆蓋座位狀態
 7：查詢電影票資訊
 8：刪除電影票
 9：清空電影院的所有座位
10：刪除電影院
11：刪除所有資料
12：停止本程式
13：教學
14：轉換配色
15：轉換語言
16：清除屏幕
請選擇模式 ([0-9]|1[0-6])
->
```

Login page in traditional Chinese

```
電影售票系統



請登錄為管理員以初始化系統。



用戶名稱：
```

User mode in Chinese:

```
電影售票系統




0：登出
1：買票
2：查票
3：退款
4：教學
請選擇模式 ([0-4])：
-> 
```

Basically, all menus and messages support traditional Chinese. However, user/admin still has to type English for input that requires special format (e.g., command in admin mode `6` (`Seat status override` / 覆蓋座位狀態)).

```
請選擇模式 ([0-9]|1[0-6])
-> 6
注意：任何透過本模式對座位狀態的更改都不會更動電影票
職員在覆蓋座位狀態前，應該先自行查詢原來的座位狀態
本模式不會幫你檢查原來的座位狀態
指令格式:

(EMPTY | BUY | RESERVE) - <電影院號碼> - <坐標表達式>
(或按 Enter 以返回控制面板)
```

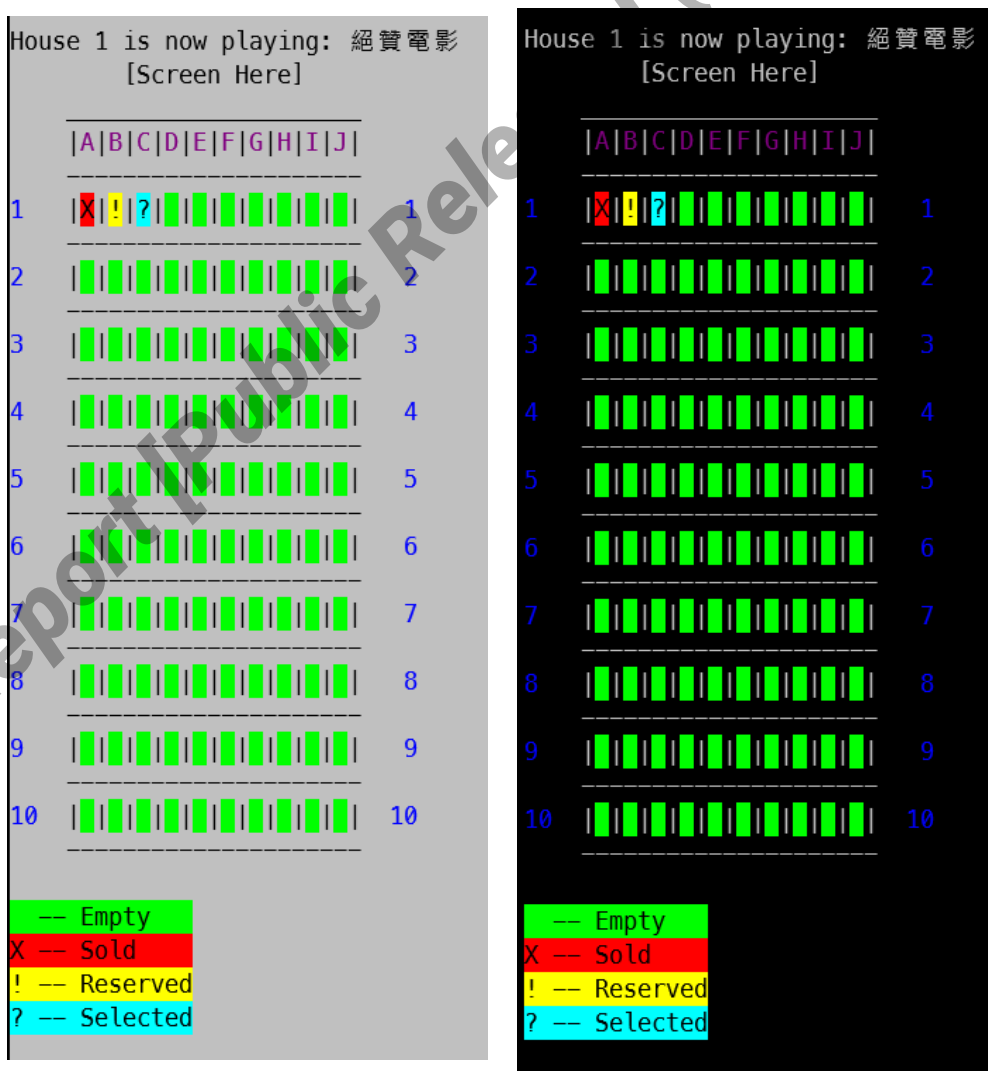
In the above example, admin has to type the command in English, including the coordinates.

As mentioned in the status storage part, this program stores the language option at `SBA/data/language.txt`. A word (‘`ENGLISH`’ or ‘`CHINESE`’) is used to indicates the language option. English is the default option.

4.13 Colour scheme

This program uses ANSI escape sequences to print characters in colour and to set background colour. Note that different terminal software (CMD / PowerShell / Terminal / GNOME Terminator) may have different colour presentation with the same ANSI escape code. This program support switching between white text on black background (*dark*)and black text on white background (*light*).

For example, here are two screenshots showing the same seating plan in light and dark mode respectively:



Same as the language option, only administrator has the permission to change the colour scheme. To do so, login as administrator, enter mode `14` (`Change the colour scheme`). As mentioned in the status storage part, this program stores the language option at `SBA/data/colour.txt`. A word (‘`LIGHT` or ‘`DARK`) is used to indicates the colour scheme. Dark mode is the default option.

Due to the fact that ANSI escape sequences are frequently used in the program. `Colour` class is used to encapsulate different sequences as class attributes of `Colour`. Inheritance from the built-in `enum.Enum` class was considered but not implemented as I don’t think an enumeration class is needed for such a simple encapsulation. Here is the code of `Colour`:

```
class Colour:
    RESET: str = '\033[0m'
    BLUE: str = '\033[1;94m'  # For row letters
    PURPLE: str = '\033[1;35m'  # For column numbers
    RED: str = '\033[1;31m'
    BLACK: str = '\033[30m'
    GREEN: str = '\033[1;32m'
    WHITE: str = '\033[37m'
    CYAN_BG: str = '\033[106m'
    WHITE_BG: str = '\033[47m'
    GREEN_BG: str = '\033[102m'
    YELLOW_BG: str = '\033[103m'
    RED_BG: str = '\033[101m'
    BLACK_BG: str = '\033[40m'
```

4.14 Error handling

Data validation, data verification and error handling are everywhere in this program.

From the viewpoint of the whole main program `main()`, it is inside a `try/except` block which handles all exception, here are the possible errors/exceptions:

Case 1: Admin Exit

Administrator is able to stop the entire program. To do so, login as administrator, enter mode `12` (`STOP THE ENTIRE PROGRAM`), as expected, the program will be stopped successfully.

```
Please choose a mode ([0-9]|1[0-6])
-> 12
See you later!
Admin Exit -- Bye Bye! \(●'◡'●)/
```

This is done by calling `quit()` to raise `SystemExit`, the try/except block then also call `quit()` to raise `SystemExit` again. So that it will really quit even in the recursion caused in the below cases.

Case 2: User / Admin pressing CTRL+C

If user or admin pressed CTRL+C, which is a special keystroke that will raise `KeyboardInterrupt`, the program will ask for confirmation, if user/admin really wants to quit, enter 'Y'. If not, enter 'N', the program will call `main()` again, causing a recursion. For example, pressing CTRL+C when inputting username:

```
CINEMA KIOSK SYSTEM


To initialize, please login as admin.



Username:

You just pressed a special keystroke.
You are going to quit this program. Are you sure about that? [y/N]Y
Got it. Have a nice day.
```

Case 3: Program Forced Exit

If the Python version is older than 3.11, the program raises `ProgramForcedExit` (inherited from `Exception`), such an exception really needs to stop the program immediately, just like an Admin Exit, but Program Forced Exit requires extra handling.

Case 4: Unexpected Errors

The program will run `main()` again, causing a recursion.

4.15 Log files

Log files are `.txt` files stored in `SBA/logs`. Logs cannot be viewed and deleted by anyone in the program. It is expected that log files can:

1. help back-end staff or manager to analyze user/administrator behaviors (for marketing research and statistic purpose and so on); and
2. help program maintainer / developer to debug and improve this program (for example, improving UI/UX and program).

Log files in this program rarely show user inputs directly. It is intended to keep the focus of the log file to the two purposes above. Plus, recording and tracking every single user input is very creepy. Note that log files are not database, they should not be used to check or review large amounts of information/data (such as houses and tickets information).

The name of the log files is the time of starting the program. Even `main()` has been called more than once due to error handling, there will only be one single log file for each execution. In the beginning of the file, before log messages, some basic information of the execution (e.g., execution time and path) will be logged.

Format of log messages

The format of a log message is:

```
%(asctime)s --> %(levelname)s @%(name)s --> %(message)s
```

Where `%(asctime)s` is the time, `%(levelname)s` is the log level, `%(name)s` is where the log message was sent, `%(message)s` is the log message. Most of the log messages should be, and are designed to be, very intuitive and straight forward.

(P.S. In most of the circumstances, `%(name)s` is the function / method name where the log message was sent.)

4.16 Data validation

Many inputs from admin/user are used after data validation, which ensures the accuracy of the input content, and can prevent operations by mistakes. For example, refer to part **4.7** (about getting input as a coordinate expression) and part **4.10** (about getting input as a ticket number), *presence check*, *length check*, *format check* and *type check* are used to ensure that the input is valid. If not, the program will return to the menu.

4.17 Testing

Unit tests

To ensure some basic components always work properly, unit tests are involved in developing this program. I use the built-in `unittest` module and the third-party `pytest` framework to test the `House` class, `coorutils.coorExprAnalysis()` and `coorutils.getCoorFromCoorExpr()`.

For example, the code below ensures that `coorExprAnalysis('A1')` will always return `[(0, 0)]`:

```
class Test_coorExprAnalysis(TestCase):
    def test_singleCoordinate(self):
        self.assertEqual(coorExprAnalysis('A1'), [(0, 0)])
```

And the code below ensures that `coorExprAnalysis('7Y::6C')` will always raise `MoreThanOneColon` exception:

```
class Test_coorExprAnalysis(TestCase):
    def test_multipleInvalidSyntax(self):
        with self.assertRaises(MoreThanOneColon):
            coorExprAnalysis('7Y::6C')
```

After executing `pytest`, the result will be printed on screen like this:

```
============================= test session starts ==============================
collecting ... collected 12 items

tests/test_coorutils.py::Test_coorExprAnalysis::test_functionException PASSED [  8%]
tests/test_coorutils.py::Test_coorExprAnalysis::test_multipleInvalidSyntax PASSED [ 16%]
tests/test_coorutils.py::Test_coorExprAnalysis::test_rangeCheck PASSED   [ 25%]
tests/test_coorutils.py::Test_coorExprAnalysis::test_singleCoordinate PASSED [ 33%]
tests/test_coorutils.py::Test_coorExprAnalysis::test_singleInvalidSyntax PASSED [ 41%]
tests/test_coorutils.py::Test_coorExprAnalysis::test_stringFormatting PASSED [ 50%]
tests/test_coorutils.py::Test_coorExprAnalysis::test_twoCoordinates
tests/test_coorutils.py::Test_getCoorsFromCoorExpr::test_multipleCoordinates PASSED [ 66%]
tests/test_coorutils.py::Test_getCoorsFromCoorExpr::test_singleCoordinate PASSED [ 75%]
tests/test_house.py::Test_House::test_initHouse PASSED                   [ 83%]
tests/test_house.py::Test_House::test_seatingPlanOperation PASSED        [ 91%]
tests/test_house.py::Test_House::test_ticket PASSED                      [100%]PASSED [ 58%]

============================== 12 passed in 0.28s ==============================
```

Static type checking

With type hints in Python, the type of variables and arguments can be ensured to have the correct type, which enhance the code readability and maintainability. Due to the fact that Python is a dynamically typed language, a third-party static type checker, `mypy`, is used.

```
mypy: install_package> python -I -m pip install --force-reinstall --no-deps D:\Programming\SBA\
 .tox\.tmp\package\295\SBA_JoeChau-1.0.tar.gz
mypy: commands[0]> mypy src
Success: no issues found in 11 source files
.pkg: _exit> python D:\Programming\SBA\venv\Lib\site-packages\pyproject_api\_backend.py True
setuptools.build_meta
```

Style Check

PEP 8 – Style Guide for Python Code is a style guideline for Python code, this is adapted by many real-world projects. *PEP 8* recommends widely accepted naming conventions and other coding conventions that would enhance the code readability and consistency. In this project, *PEP 8* is adapted with a few modifications. The overall naming style in this project is:

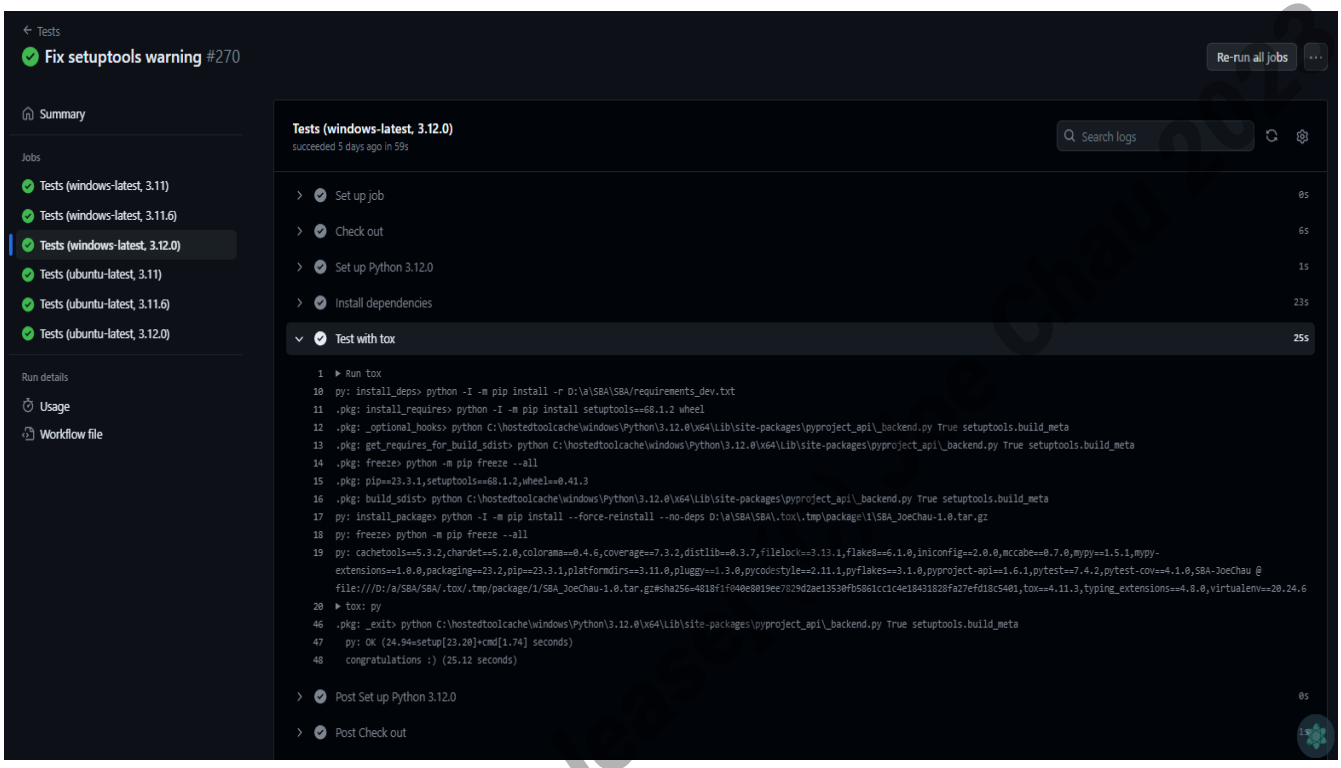
- `lower_case_with_underscores` for variables
- `UPPER_CASE_WITH_UNDERSCORES` for constants and important variables
- `mixedCase` for functions and methods
- `CapitalizedWords` for classes
- `i` (single lowercase letter) for temporary variables used in iterations and loops

`flake8`, a third-party library, is used to check the code style.

```
flake8: install_package> python -I -m pip install --force-reinstall --no-deps D:\Programming\SBA\.tox\
 .tmp\package\294\SBA_JoeChau-1.0.tar.gz
flake8: commands[0]> flake8 src tests
flake8: OK ✓ in 6.06 seconds
```

Tests Automation

`tox`, another third-party module, is used to integrate these three tests. Moreover, I use GitHub Actions to automatically do these tests when I push my code into the GitHub repository. Here is a screenshot of it:



4.18 Miscellaneous

Documentation

There is an online documentation https://joeccp.github.io/SBA/ (hosted with GitHub Pages), providing similar documentation of this report for this program. User/Admin can see the documentation with admin mode `13` (`Help`) or user mode `4` (`HELP`). The program will automatically open the website with a browser (with the help of the built-in `webbrowser` module).

Here is a screenshot of the website:

Clear screen

Unlike user mode, admin Control Panel will not clear the screen every time an operation is done. To do so, admin can enter admin mode `16` (`Clear screen`). The language option and colour scheme will not be changed due to clear screen. The menu of the Control Panel will be printed again after clear screen.

Code documents

The code of the program can be view at https://github.com/Joeccp/SBA or https://joeccp.github.io/SBA/codedocs/src/index.html (generated with `pdoc`, a third-party CLI tool).

License

This project is open-source, it is released under the *Apache License, Version 2.0*, the license can be obtained at https://www.apache.org/licenses/LICENSE-2.0.

Packaging

This project is packaged as a module, as mentioned in part **4.2** end user can execute this program with the command `python SBA`, so that end user need not to know exactly which script to be run. Python will find the `__main__.py` inside the `SBA` folder and execute it.

5 Acknowledgement

Type	Item	Copyright holder	URL	Remarks
Image	`SBA/images/site/favicon.ico`	Material Symbols by Google	https://fonts.google.com/icons	Licensed under the Apache License, Version 2.0
File Template	`SBA/.gitignore`	GitHub	https://github.com/github/gitignore	Licensed under Creative Commons Zero v1.0 Universal.

File Template	`SBA/.github/workflows/tests.yml`
	`SBA/pyproject.toml`
	`SBA/setup.cfg`
	`tox.ini`
		mCoding	https://www.youtube.com/watch?v=DhUpxWjOhME
			https://github.com/mCodingLLC/SlapThatLikeButton-TestingStarterProject
				Licensed under the MIT License

I would like to express a token of thanks to the following individuals (ordered alphabetically):

1. [redacted]
2. [redacted]
3. [redacted]

Thank you for their guidance and inspiration!

6 Appendix

6.1 Source code

The source code may be seen at https://github.com/Joeccp/SBA.