Question 1

Learning objectives:

- Use objects to store and compute tabular information,
- Apply the principles of object-oriented programming principles in designing and developing applications

This question is about newspaper subscription for either a print or an online version.

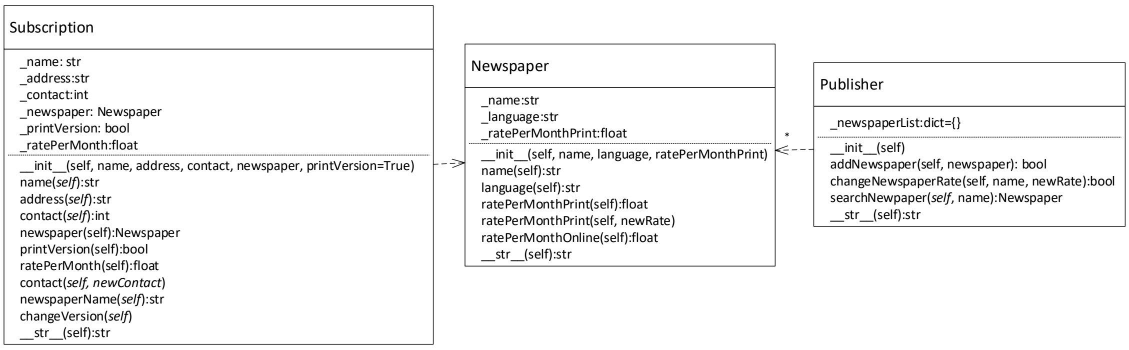


Figure Q1

(a) The Newspaper class models one newspaper that can be subscribed. Assume that both printed and online versions are available. The class has the following attributes:

- _name – An instance variable for the name of a newspaper.
- _language – An instance variable for the language the newspaper articles are written in.
- _ratePerMonthPrint – An instance variable for the monthly subscription rate for a printed version of the newspaper.
- __init__ - The method initialises a new newspaper.
- Getter methods for all instance variables and a setter method for _ratePerMonthPrint.
- ratePerMonthOnline – The method returns the monthly subscription rate for an online version of the newspaper which is 50% the rate for a printed version.
- __str__ – The method returns a string representation of a newspaper. E.g.,

```
Name: Today's News    Language: English   Current Rate Print: $34.90/mth Online: $17.45/mth
```

(i) Define the Newspaper class.
Write a function named Q1a() and include statements for these steps:

(ii) Create these Newspaper objects as shown in Table Q1(a).

	Name	Language	Rate Per Month Print
1	Today's News	English	$34.90
2	Berita Hari	Malay	$45.90
3	Zao Pao	Chinese	$47.50
4	Inru ceyti	Tamil	$47.90

Table Q1(a)

(iii) Increase the monthly subscription rate for print version of the newspaper with the name Today's News by $5, and then display the new rate.

(iv) Display name and language of the second newspaper and the details of the third newspaper.

(5 marks)

(b) The Subscription class models one subscription for a version (print or online) of a newspaper. It has the following attributes:

- _name, _address and _contact – Three instance variables for the name, address and contact of a person making the subscription.
- _newspaper – An instance variable for the newspaper the subscription is for.
- _printedVersion – An instance variable for whether the subscription is for a printed version (True) or for an online version (False).
- _ratePerMonth – An instance variable which records the monthly subscription rate of the newspaper based on the version and the newspaper.
  When the monthly subscription rate changes, the change affects only new subscriptions.
- __init__ - The method initialises a new subscription. The parameter printVersion has a default value True.
- Getter methods for all instance variables and a setter method for contact.
- newspaperName – The method returns the name of the newspaper the subscription is for.
- changeVersion – The method changes the newspaper version that the subscription is for. If the version is printed, _printedVersion changes from True to False (online). If the version is online, _printedVersion changes from False to True (printed). _ratePerMonth is adjusted accordingly, based on the prevailing subscription rate for the newspaper.
- __str__ – The method returns a string representation of a subscription. Two examples are shown here:

  Example 1: Subscription for online version

```
Peter 42 Simei Drive 96667878 Print Version:No  Subscription Rate: $17.45 per month
Name: Today's News    Language: English   Current Rate Print: $34.90/mth Online: $17.45/mth
```

  Example 2: Subscription for printed version

```
John 12 Clementi Road 98769876 Print Version:Yes Subscription Rate: $34.90 per month
Name: Today's News    Language: English   Current Rate Print: $34.90/mth Online: $17.45/mth
```

(i) Define the Subscription class.

Write a function named Q1b() and include statements for these steps:

(ii) Create **TWO (2)** Subscription objects for Today's New currently at $34.90 for printed version.

	Name	Address	Contact	Newspaper	PrintVersion
1	Peter	42 Simei Drive	96667878	1	False
2	John	12 Clementi Road	98769876	1	True

Table Q1(b)

(iii) Display the detail of subscription 1 and the name of newspaper for subscription 2.

(iv) Increase the monthly subscription rate for the newspaper for subscription 1 for print version to 37.50, and then change the version for subscription 1

(v) Display the rates for both subscriptions and the prevailing subscription rates for printed and online versions for newspaper that the subscriptions are for.

(7 marks)

(c) The Publisher class models a publisher of newspapers. It has the following attributes:

- _newspaperList – An instance variable for a list of newpaper published by the publisher. _newspaperList is a dictionary with the name of the newspaper as key and the newspaper as value.
- __init__ – The method initialises the newspaper list as an empty dictionary.
- addNewspaper – The method adds a newspaper to _newspaperList.
- changeNewspaperRate – The method searches for a newspaper in _newspaperList that has the same name as the parameter name, and changes the subscription rate for that newspaper. The method returns True if the search is successful and the change is made, Otherwise, the method returns False.
- searchNewspaper – The method returns a newspaper in _newspaperList with a name that matches the parameter name. The method returns None otherwise.
- __str__ – The method returns the dictionary of newspaper as a string. E.g.,

```
Name: Today's News    Language: English   Current Rate Print: $34.90/mth Online: $17.45/mth
Name: Berita Hari     Language: Malay     Current Rate Print: $45.90/mth Online: $22.95/mth
Name: Zao Pao         Language: Chinese   Current Rate Print: $45.90/mth Online: $22.95/mth
Name: Inru ceyti      Language: Tamil     Current Rate Print: $46.90/mth Online: $23.45/mth
```

(i) Define the Publisher class.

Write a function named Q1c() and include statements for these steps:

(ii) Create a Publisher object and add all 4 newspapers shown in Table Q1(a).

(iii) Search for the newspaper Zao Pao. If it can be located, display its rates for both printed and online versions.

(iv) Get the publisher to change the price for the newspaper Inru ceyti to 46.50. Then, display the publisher.

(8 marks)

Carpark Parking application for Question 2 and 3

The application tracks and computes charges for motorbikes and cars parking in two types of carparks – outside central area and within central area which incur higher charges. Different charges apply at different periods of a day, e.g., 7am to 5pm, 5pm to 10.30pm and 10.30pm to 7am. For simplicity, there is no need to differentiate between weekdays and weekends. Motorbikes are charged on per-period basis whereas cars are charged in blocks of 30 minutes or part of. Refer to Figure Q2 for the class diagram.

In Question 2, you will implement these classes:

- Carpark and CentralAreaCarpark
- ParkingCalculator, MotorbikeParkingCalculator and CarParkingCalculator

In Question 3, you will implement these classes:

- The exception classes ParkingException and ParkingDetailException
- Parking, MotorbikeParking and CarParking
- CarParkTrackingSystem

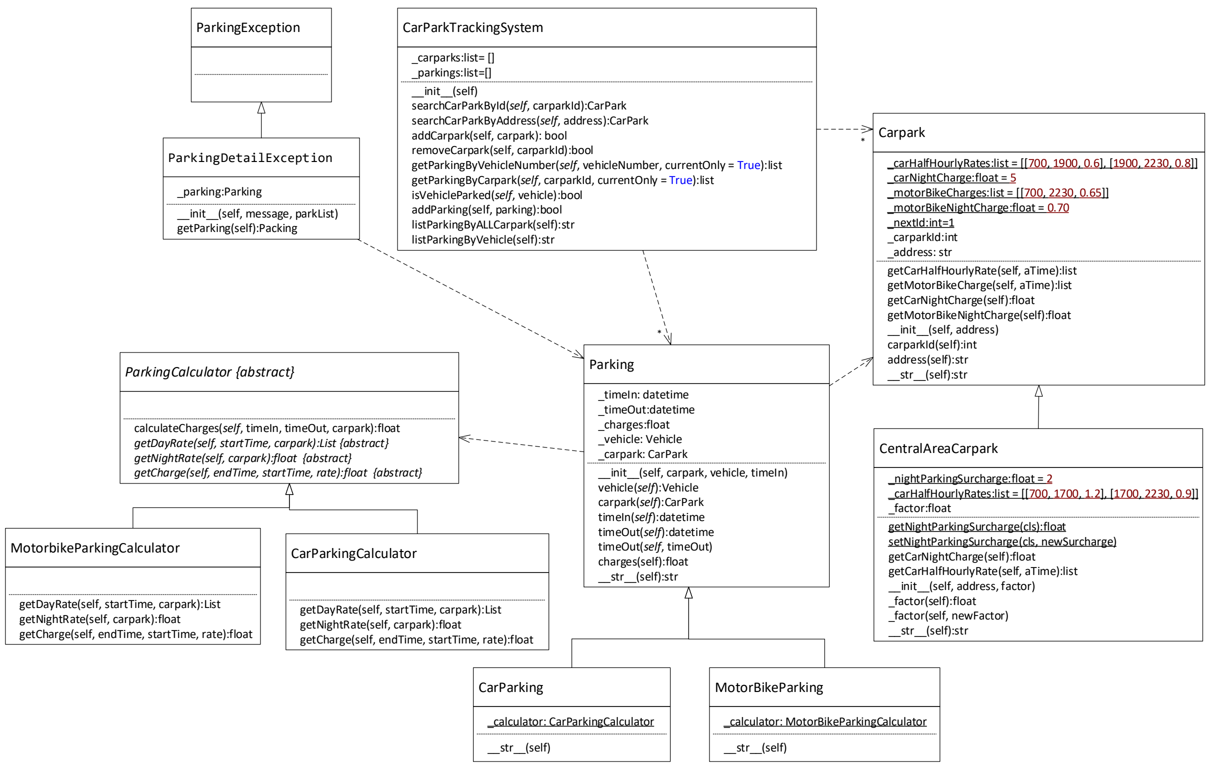

ParkingException
ParkingDetailException
_parking:Parking
__init__(self, message, parkList)
getParking(self):Packing
CarParkTrackingSystem
_carparks:list= []
_parkings:list=[]
__init__(self)
searchCarParkById(self, carparkId):CarPark
searchCarParkByAddress(self, address):CarPark
addCarpark(self, carpark): bool
removeCarpark(self, carparkId):bool
getParkingByVehicleNumber(self, vehicleNumber, currentOnly = True):list
getParkingByCarpark(self, carparkId, currentOnly = True):list
isVehicleParked(self, vehicle):bool
addParking(self, parking):bool
listParkingByALLCarpark(self):str
listParkingByVehicle(self):str
*
Carpark
_carHalfHourlyRates:list = [[700, 1900, 0.6], [1900, 2230, 0.8]]
_carNightCharge:float = 5
_motorBikeCharges:list = [[700, 2230, 0.65]]
_motorBikeNightCharge:float = 0.70
_nextId:int=1
_carparkId:int
_address: str
getCarHalfHourlyRate(self, aTime):list
getMotorBikeCharge(self, aTime):list
getCarNightCharge(self):float
getMotorBikeNightCharge(self):float
__init__(self, address)
carparkId(self):int
address(self):str
__str__(self):str
CentralAreaCarpark
_nightParkingSurcharge:float = 2
_carHalfHourlyRates:list = [[700, 1700, 1.2], [1700, 2230, 0.9]]
_factor:float
getNightParkingSurcharge(cls):float
setNightParkingSurcharge(cls, newSurcharge)
getCarNightCharge(self):float
getCarHalfHourlyRate(self, aTime):list
__init__(self, address, factor)
_factor(self):float
_factor(self, newFactor)
__str__(self):str
*
ParkingCalculator {abstract}
calculateCharges(self, timeIn, timeOut, carpark):float
getDayRate(self, startTime, carpark):List {abstract}
getNightRate(self, carpark):float {abstract}
getCharge(self, endTime, startTime, rate):float {abstract}
MotorbikeParkingCalculator
getDayRate(self, startTime, carpark):List
getNightRate(self, carpark):float
getCharge(self, endTime, startTime, rate):float
CarParkingCalculator
getDayRate(self, startTime, carpark):List
getNightRate(self, carpark):float
getCharge(self, endTime, startTime, rate):float
Parking
_timeIn: datetime
_timeOut:datetime
_charges:float
_vehicle: Vehicle
_carpark: CarPark
__init__(self, carpark, vehicle, timeIn)
vehicle(self):Vehicle
carpark(self):CarPark
timeIn(self):datetime
timeOut(self):datetime
timeOut(self, timeOut)
charges(self):float
__str__(self):str
CarParking
_calculator: CarParkingCalculator
__str__(self)
MotorBikeParking
_calculator: MotorBikeParkingCalculator
__str__(self)

Figure Q2

Question 2

Learning objectives:

- Demonstrate how class hierarchy and association can be used to organize information.
- Construct the class hierarchy and association according to specification.

(a)

(i) Define the CarPark class. The CarPark class models one carpark located outside central area. It has the following attributes:

- _carHalfHourlyRates – A class variable implemented as a nested list.
  Each element in the nested list is a list in this form: [a, b, c] where a and b are the start and end of a time period and c is the rate per 30 minutes of parking.
  For example, [[700, 1900, 0.6], [1900, 2230, 0.8]] records 2 time periods:
  - 7am to 7pm at 60 cents for each 30-minute block or part of, and
  - 7pm to 10.30pm at 80 cents for each 30-minute block or part of.
- _carNightCharge – A class variable to record the night parking charge from 10.30pm to 7am for a car is a flat rate of $5.
- _motorBikeCharges – A class variable implemented as a nested list, similar to _carHalfHourlyRates with each element in the form [a, b, c] for a motorbike, and c is the per period charge for the period between a and b. Motorbike parking is charged on a per period basis.
- _motorBikeNightCharge– A class variable to record the night parking charge from 10.30pm to 7am for a vehicle type motorbike. A motorbike is charged 70 cents for the whole period from 10.30pm to 7am.
- _nextId – A class variable to auto-generate running numbers for carpark ids.
- _carparkId - An instance variable for the id of the carpark. A carpark id identifies a car park. Carpark id is assigned using _nextId.
- _address - An instance variable for the address of the carpark.
- __init__ – The method initialises a new carpark.
- Getter methods for both instance variables.
- getCarHalfHourlyRate – The method returns a list from _carHalfHourlyRates based on the parameter aTime in the form [a, b, c] such that aTime is at least the start time, a of a time period and is less than the end time of a time period, b.
  For example, if aTime is 2000, then the list [1900, 2230, 0.8] is returned.
  The method returns None if there is no time period for aTime.
- getMotorBikeCharge – The method does the same task as getCarHalfHourlyRate except that method uses the class variable _motorBikeCharges.
- getCarNightCharge – The method returns the night parking charge for the vehicle type car.
- getMotorBikeNightCharge – The method returns the night parking charge for the vehicle type motorbike.
- __str__ – The method returns a string representation of the carpark. E.g.,

```
Id: 1 11A Clementi Ave 12 **Outside of Central Area**
```

(ii) Define the CentralAreaCarPark class. The CentralAreaCarPark class models one carpark within the central area. The CentralAreaCarPark class is a subclass of the CarPark class, and has these attributes:

- _factor – An additional instance variable, factor records how centralised the carpark is. _factor is used to adjust the rate of parking.
- _nightParkingSurcharge – A class variable for a surcharge for night parking for car in a carpark in central area.
- _carHalfHourlyRates – A class variable which replaces the superclass' variable _carHalfHourlyRates.
- A getter and setter method for _nightParkingSurcharge
- getCarNightCharge – The method adds the surcharge for night parking to the night charge for the superclass.
- getCarHalfHourlyRate – The method adjusts the rate by multiplying it with the factor before returning the list.
- __init__ – The method initialises a new central area car park.
- Getter and setter methods for factor.
- – a getter method to return and setter method to update the minimum age.
- __str__ – The method returns a string representation of a central area carpark. E.g.,

```
Carpark Id: 2 Factor: 1.1 123 Syed Ameen Road **Within Central Area**
```

(12 marks)

(b) The ParkingCalculator class is an abstract superclass to the classes MotorbikeParkingCalculator and CarParkingCalculator. These classes implement the algorithm to compute parking changes based on the type of carpark.

(i) Define the ParkingCalculator class. The ParkingCalculator class models a calculator for car parking and has the following attributes:

- calculateCharge – The method computes and returns parking charge based on the time the vehicle enters the carpark (timeIn), the time the vehicle leaves the carpark (timeOut) and the type of carpark.
  If the vehicle is in the carpark for l0 minutes or less, there is no charge for the 10 minutes. If the duration is more than 10 minutes, the whole duration inclusive of the first 10 minutes is chargeable.
  The method repeatedly gets the day rate and/or night rate for a period, and uses the retrieved rate to get the charge for a period. The charges for the periods between timeIn and timeOut are accumulated and returned.
  For example, a vehicle parks from 20 Sep 2020, 21:29 to 21 Sep 2020, 07:36, and the rates per 30-minute block are [[700, 1900, 0.6], [1900, 2230, 0.8]] with night charges at $5 from 2230 to 700 the next day. These steps are performed to compute the parking charges:
  - Retrieve the (day) rate [1900, 2230, 0.8] for the time 21:29
  - Compute the number of 30 minutes block between 2129 and 2230 which is 3. The charge for this period is 3 * 0.8 = $2.40
  - Retrieve the (night) rate for the time 2230. Night charge of $5 applies.
  - Retrieve the (day) rate [700, 1900, 0.6] for the time 7:00

    - Compute the number of 30-minute block from 7:00 to 7:36 which is 2. The charge for this period is 2 * 0.6 = $1.20.
    - Add the charges for all the periods vehicle is parked = $2.40 + $5 + $1.20 = $8.60
- getDayRate, getNightRate and getCharges are abstract methods.

(ii) Define MotorbikeParkingCalculator class. The MotorbikeParkingCalculator class models a calculator for motorbike parking and implements all the abstract methods:

- getDayRate – The method gets the day rate for motorcycle for a given time from the carpark.
- getNightRate – The method gets the night rate for motorcycle from the carpark.
- getCharges – The method returns the charges for parking for a given duration between the start time and the end time.

(iii) Define CarParkingCalculator class. The CarParkingCalculator class models a calculator for car parking and implements all the abstract methods:

- getDayRate – The method gets the day rate for car for a given time from the carpark.
- getNightRate – The method gets the night rate for car from the carpark.
- getCharges – The method returns the parking charge for the period which is the product of the rate and the number of 30-minute block or any part of.

(10 marks)

Question 3

Learning objectives

- Apply the principles of object-oriented programming principles in designing and developing applications,
- Construct the class hierarchy and association according to specification

(a) Define the exception classes.

(i) Define a subclass, ParkingException of the Exception class. The class ParkingException has no additional attribute.

(ii) Define a subclass ParkingDetailException of the ParkingException class. The class has an additional instance variable, _parking and a getter method. _parking will refer to the parking that causes the exception.

When the application encounters a business rule violation, an exception from one of these two classes is raised.

(3 marks)

(b) The Parking class is an abstract superclass to the class MotorBikeParking and the class CarParking. The Parking class models one parking for one vehicle, entering in one carpark at a specific (time in) and leaving after some time (time out). The Parking class is also associated with an appropriate ParkingCalculator for computing the parking charges.

(i) Define the Parking class as an abstract class. The Parking class has the following attributes:

- _calculator – A class variable that records the parking calculator required. _calculator is set to None as Parking class is abstract and it does not calculate parking charges.
- _timeIn – An instance variable for the time the vehicle enters the carpark.
- _timeOut – An instance variable for the time the vehicle leaves the carpark.
- _charges – An instance variable for the charges incurred for the parking, for accounting purposes as parking rates can change over time but parking charges, once computed, will not change.
- _vehicleNumber – An instance variable for the vehicle number of vehicle for this parking.
- _carpark – An instance variable for the carpark for this parking.
- __init__ – The method initialises a parking with the provided parameter values. _timeOut is assigned to None and _charges is set as 0.
- Getter methods for all instance variables. If _timeOut is None, the getter method for _charges raises a ParkingException with the exception message: Charges not recorded yet as vehicle has not left carpark.
- The setter method for _timeOut raises a ParkingException if
  - _timeOut is not None. The exception message is: Time out has already been recorded!
  - the parameter value for timeOut is earlier _timeIn. The exception message is: Time out cannot be earlier than time in!

  Otherwise, it sets _timeOut, calls the calculator of the vehicle to calculate the charges, and set _charges with the return value

- __str__ – The method returns a string representation of a parking in various formats. Three examples are shown here:
  - Example 1: A parking where a vehicle leaves within 10 minutes.
    ```
    Time In: 15 Sep 2020, 10:29 PM Time Out: 15 Sep 2020, 10:39 PM Charges: $0.00
    Carpark Id: 1 11A Clementi Ave 12 **Outside of Central Area**
    Vehicle: SKY1234G
    ```
  - Example 2: A parking where a vehicle has not left.
    ```
    Time In: 15 Sep 2020, 11:55 PM Time Out: *Parked in carpark 2  Charges: $0.00
    Carpark Id: 2 Factor: 1.1 123 Syed Ameen Road **Within Central Area**
    Vehicle: GGU4567N
    ```
  - Example 3: A parking where a vehicle has left
    ```
    Time In: 15 Sep 2020, 05:29 PM Time Out: 15 Sep 2020, 10:36 PM Charges: $13.00
    Carpark Id: 1 11A Clementi Ave 12 **Outside of Central Area**
    Vehicle: SKY99G
    ```

(ii) Define MotorbikeParking class

- _parkingCalculator – A class variable for a parking calculator which computes parking charges for motor bike parking.
  Set the variable to a MotorBikeParkingCalculator object.
- __str__ – The method returns a string representation of a car parking. It simply appends the string `Motor Bike` to the output of the __str__ method of its superclass. E.g.,

```
Time In: 15 Sep 2020, 11:55 PM Time Out: *Parked in carpark 2  Charges: $0.00
Carpark Id: 2 Factor: 1.1 123 Syed Ameen Road **Within Central Area**
Vehicle: GGU4567N Motor Bike
```

(iii) Define CarParking class

- _parkingCalculator – A class variable for a parking calculator which computes parking charges for car parking.
  Set the variable to a CarParkingCalculator object.
- __str__ – The method returns a string representation of a car parking. It simply appends the string `Car` to the output of the __str__ method of its superclass. E.g.,

```
Time In: 15 Sep 2020, 10:29 PM Time Out: 15 Sep 2020, 10:39 PM Charges: $0.00
Carpark Id: 1 11A Clementi Ave 12 **Outside of Central Area**
Vehicle: SKY1234G Car
```

(10 marks)

(c) Define the CarParkTrackingSystem which keeps track of carparks and parking. The CarParkTrackingSystem class has the following attributes:

- Two instance variables maintain a list of carparks and a list of parking.
- __init__ – The method initialises a carpark tracker with two empty collections.
- There are two search methods to search for a carpark:
  - searchCarParkById has a parameter carparkId and
  - searchCarParkByAddress has a parameter address,

  The method returns the carpark with the matching parameter value, if the carpark is located. Otherwise the method returns None.

- addCarpark – The method searches the collection of carparks by address, and if the carpark with the address is not in the collection yet, then the carpark is added, and the method returns True. Otherwise, the method returns False.
- getParkingByVehicleNumber – If currentOnly is False, the method returns a list of all parking for a vehicle with the parameter vehicleNumber. If currentlyOnly is True, the method returns a list of all parking for a vehicle still in a carpark.
- getParkingByCarpark– If currentOnly is False, the method returns a list of parking for a carpark with the parameter carparkId. If currentlyOnly is True, the method returns a list of parking for vehicles still in the carpark.
- addParking – The method adds the parameter parking to the list of parking if the vehicle for the parking has left the carpark for all previous parking, and the method returns True. Otherwise, the method raises a ParkingDetailException with the offending parking where the exit time has not been recorded, together with the

exception message: Error in adding a parking. Parking record shows vehicle is still in a carpark.

- listParkingByAllCarpark – The method returns a string consisting of parking details in each carpark in the collection, and a summarised detail of the total collected and the number of vehicles that are still in the carparks. An example here shows two carparks being tracked.

```
Detail of carpark Id: 1 11A Clementi Ave 12 **Outside of Central Area**
Time In: 15 Sep 2020, 10:29 PM Time Out: 15 Sep 2020, 10:39 PM Charges: $0.00
Carpark Id: 1 11A Clementi Ave 12 **Outside of Central Area**
Vehicle: SKY1234G Car
Time In: 15 Sep 2020, 10:29 PM Time Out: 16 Sep 2020, 07:15 AM Charges: $6.40
Carpark Id: 1 11A Clementi Ave 12 **Outside of Central Area**
Vehicle: SKY1234G Car
Time In: 15 Sep 2020, 08:29 PM Time Out: *Parked in carpark 1  Charges: $0.00
Carpark Id: 1 11A Clementi Ave 12 **Outside of Central Area**
Vehicle: GGN1234J Motor Bike
Number of completed parking: 2
Number of vehicles parked in carpark currently: 1
Total amount: $6.40

Detail of carpark Id: 2 Factor: 1.1 123 Syed Ameen Road **Within Central Area**
Time In: 15 Sep 2020, 01:29 PM Time Out: 15 Sep 2020, 03:29 PM Charges: $5.28
Carpark Id: 2 Factor: 1.1 123 Syed Ameen Road **Within Central Area**
Vehicle: SKY1234G Car
Time In: 15 Sep 2020, 08:45 AM Time Out: 15 Sep 2020, 10:29 AM Charges: $5.81
Carpark Id: 2 Factor: 1.1 123 Syed Ameen Road **Within Central Area**
Vehicle: SKU9876X Car
Time In: 15 Sep 2020, 08:45 PM Time Out: 16 Sep 2020, 10:30 PM Charges: $2.00
Carpark Id: 2 Factor: 1.1 123 Syed Ameen Road **Within Central Area**
Vehicle: GGU4567N Motor Bike
Time In: 15 Sep 2020, 05:29 PM Time Out: 15 Sep 2020, 10:36 PM Charges: $17.89
Carpark Id: 2 Factor: 1.1 123 Syed Ameen Road **Within Central Area**
Vehicle: SKY99G Car
Number of completed parking: 4
Number of vehicles parked in carpark currently: 0
Total amount: $30.98

Summary for all carparks
Total collected: $37.38
Number of vehicles parked in all carparks currently: 1
```

(10 marks)

(d) Write the carpark tracking application that creates a CarParkTrackingSystem object, populates it with the list as shown under the method listParkingByAllCarpark in Q3c, and then provides the following menu option:

```
Menu
1. Park a Vehicle
2. Exit Carpark
3. Display Parking Summary for all Carparks
0. Exit 0. Exit
```

The application must handle exceptions as specified in the various menu options.

- input error such as Type or Value error are handled by allowing the user to re-enter until the value entered has the correct data type or format. E.g.,

  <u>Incorrect data type for menu option:</u>

  ```
  Menu
  1. Park a Vehicle
  2. Exit Carpark
  3. Display Parking Summary for all Carparks
  0. Exit
  Enter option: 1.1
  ```

```
The input is not a number
Menu
1. Park a Vehicle
2. Exit Carpark
3. Display Parking Summary for all Carparks
0. Exit
Enter option: xyz
The input is not a number
Menu
1. Park a Vehicle
2. Exit Carpark
3. Display Parking Summary for all Carparks
0. Exit
Enter option: 1
```

Incorrect data type for parking type:

```
Enter type of parking 1 - Car  2 - Motor bike: x
Invalid value
Enter type of parking 1 - Car  2 - Motor bike: 3
```

Incorrect datetime format:

```
Enter time in in dd/mm/yyyy hh:mm AM/PM format: 1 Oct 2020 1545
Not in correct dd/mm/yyyy hh:mm AM/PM  format
Enter time in in dd/mm/yyyy hh:mm AM/PM format: 1-10-2020 3:45PM
Not in correct dd/mm/yyyy hh:mm AM/PM  format
Enter time in in dd/mm/yyyy hh:mm AM/PM format: 1/10/2020 3:34 pm
```

- ParkingException is handled by printing the error message and returning to the main menu.
- ParkingDetailedException is handled according to the specifications of the question. For option 1, when a parking record shows that avehicle is still in a carpark, handle the ParkingDetailedException by allowing the user to decide whether update the parking record to exit the vehicle. If user chooses to exit, then the parking charges are computed, and the parking for the new entry is added.

Assume all vehicle number entered is valid.

(i) Option 1: Park a Vehicle
The application prompts the user for the type of vehicle, vehicle number and time in. If the user enters a number when prompted for a type of vehicle, and the user enters a number that is not 1 or 2, or if the user enters a number when prompted for a carpark id but the number is not a carpark id for an existing carpark, an error message is displayed, and then, the user is presented the main menu. Samples interactions are shown here (user input are shown underlined)

Incorrect value for parking type:

```
Enter type of parking 1 - Car  2 - Motor bike: 3
Parking type is not a correct parking type.
```

Incorrect value for carpark id:

```
Enter carpark id: 4
carpark id does not belong to any carpark
```

Example 1: Successful parking

```
Enter option: 1
Enter type of parking 1 - Car  2 - Motor bike: 1
Enter carpark id: 1
Enter vehicle number: SKY1234G
Enter time in in dd/mm/yyyy hh:mm AM/PM format: 16/9/2020 10:22 am
Time In: 16 Sep 2020, 10:22 AM Time Out: *Parked in carpark 1  Charges: $0.00
```

```
Carpark Id: 1 11A Clementi Ave 12 **Outside of Central Area**
Vehicle: SKY1234G Car *added.
```

Example 2: Successful parking

```
Enter option: 1
Enter type of parking 1 - Car  2 - Motor bike: 2
Enter carpark id: 2
Enter vehicle number: GGN1234J
Enter time in in dd/mm/yyyy hh:mm AM/PM format: 16/9/2020 9:35 am
Error in adding a parking. Parking record shows vehicle is still in a carpark
Time In: 15 Sep 2020, 08:29 PM Time Out: *Parked in carpark 1  Charges: $0.00
Carpark Id: 1 11A Clementi Ave 12 **Outside of Central Area**
Vehicle: GGN1234J Motor Bike
Do you want to exit the vehicle for this parking? y/n: y
charges: $2.00
Time In: 16 Sep 2020, 09:35 AM Time Out: *Parked in carpark 2  Charges: $0.00
Carpark Id: 2 Factor: 1.1 123 Syed Ameen Road **Within Central Area**
Vehicle: GGN1234J Motor Bike *added.
```

Example 3: Unsuccessful parking

```
Enter option: 1
Enter type of parking 1 - Car  2 - Motor bike: 2
Enter carpark id: 2
Enter vehicle number: GGN1234J
Enter time in in dd/mm/yyyy hh:mm AM/PM format: 16/9/2020 10:22 am
Error in adding a parking. Parking record shows vehicle is still in a carpark
Time In: 16 Sep 2020, 09:35 AM Time Out: *Parked in carpark 2  Charges: $0.00
Carpark Id: 2 Factor: 1.1 123 Syed Ameen Road **Within Central Area**
Vehicle: GGN1234J Motor Bike
Do you want to exit the vehicle for this parking? y/n: n
Choosing not to exit previous parking
```

(ii) Option 2: Exit CarPark

The application prompts the user for a vehicle number and time out, and then computes and displays the parking charges. Samples interactions are shown here (user input are shown underlined)

Example 1: Successful exit

```
Enter option: 2
Enter vehicle number: GGN1234J
Time In: 16 Sep 2020, 09:35 AM Time Out: *Parked in carpark 2  Charges: $0.00
Carpark Id: 2 Factor: 1.1 123 Syed Ameen Road **Within Central Area**
Vehicle: GGN1234J Motor Bike
Enter time out in dd/mm/yyyy hh:mm AM/PM format: 16/9/2020 10:22 am
charges: $0.65.
```

Example 2: Unsuccessful exit

```
Enter option: 2
Enter vehicle number: GGN1234J
Time In: 16 Sep 2020, 09:35 AM Time Out: *Parked in carpark 2  Charges: $0.00
Carpark Id: 2 Factor: 1.1 123 Syed Ameen Road **Within Central Area**
Vehicle: GGN1234J Motor Bike
Enter time out in dd/mm/yyyy hh:mm AM/PM format: 16/9/2020 8:22 am
Time out cannot be earlier than time in!
```

(iii) Option 3: Display Parking Summary for all Carparks

The application displays all parking in each carpark. An example output is shown below:

```
Detail of carpark Id: 1 11A Clementi Ave 12 **Outside of Central Area**
Time In: 15 Sep 2020, 10:29 PM Time Out: 15 Sep 2020, 10:39 PM Charges: $0.00
Carpark Id: 1 11A Clementi Ave 12 **Outside of Central Area**
Vehicle: SKY1234G Car
```

```
Time In: 15 Sep 2020, 10:29 PM Time Out: 16 Sep 2020, 07:15 AM Charges: $6.40
Carpark Id: 1 11A Clementi Ave 12 **Outside of Central Area**
Vehicle: SKY1234G Car
Time In: 15 Sep 2020, 08:29 PM Time Out: *Parked in carpark 1  Charges: $0.00
Carpark Id: 1 11A Clementi Ave 12 **Outside of Central Area**
Vehicle: GGN1234J Motor Bike
Number of completed parking: 2
Number of vehicles parked in carpark currently: 1
Total amount: $6.40

Detail of carpark Id: 2 Factor: 1.1 123 Syed Ameen Road **Within Central Area**
Time In: 15 Sep 2020, 01:29 PM Time Out: 15 Sep 2020, 03:29 PM Charges: $5.28
Carpark Id: 2 Factor: 1.1 123 Syed Ameen Road **Within Central Area**
Vehicle: SKY1234G Car
Time In: 15 Sep 2020, 08:45 AM Time Out: 15 Sep 2020, 10:29 AM Charges: $5.28
Carpark Id: 2 Factor: 1.1 123 Syed Ameen Road **Within Central Area**
Vehicle: SKU9876X Car
Time In: 15 Sep 2020, 08:45 PM Time Out: 16 Sep 2020, 10:30 PM Charges: $2.00
Carpark Id: 2 Factor: 1.1 123 Syed Ameen Road **Within Central Area**
Vehicle: GGU4567N Motor Bike
Time In: 15 Sep 2020, 05:29 PM Time Out: 15 Sep 2020, 10:36 PM Charges: $17.89
Carpark Id: 2 Factor: 1.1 123 Syed Ameen Road **Within Central Area**
Vehicle: SKY99G Car
Number of completed parking: 4
Number of vehicles parked in carpark currently: 0
Total amount: $30.45

Summary for all carparks
Total collected: $36.85
Number of vehicles parked in all carparks currently: 1
```

(12 mark)

Note: Remember to paste the complete code for Question 2 and Question 3 into another single table cell as instructed in the solution document template. This is in addition to pasting your code into the table cells for Question 2 and for Question 3.

Question 4

Learning objective:

- Develop Graphical User Interface (GUI) for an application based on user requirements

Write a GUI for a carpark manager that computes the parking charges at 4 cents per minute at all times for carpark A and at 3 cents per minute for carpark B. The rates apply to all types of vehicles.

The GUI maintains a dictionary, and the dictionary contains only one entry for each vehicle, either currently parking or has parked in carpark A or carpark B. Use vehicle number as key and a list of 4 items [a, b, c, d] where a and b are the time in and time out, c is the carpark and d is the parking charges. If vehicle has not exited, then b is None and d is 0. The dictionary starts as empty.

(a) Implement the GUI as closely as possible to the one shown in Figure Q4(a). The title of the GUI MUST include your name. Note that the textfields for time in and time out and the submit buttons are disabled when the GUI shows up.

(10 marks)

(b) Implement event handling for the first 5 buttons. You do not need to implement the Clear button. Furthermore, you do not need to handle exceptions for this question.

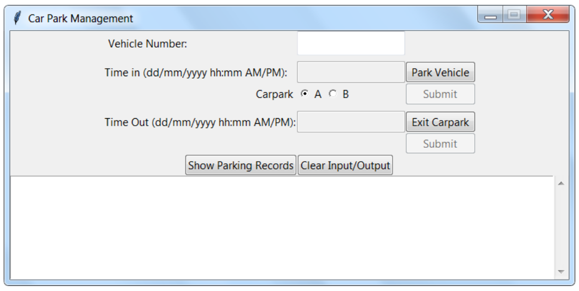
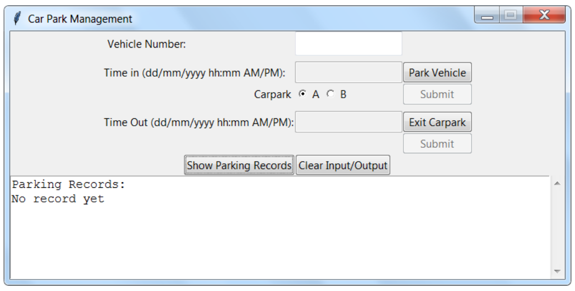
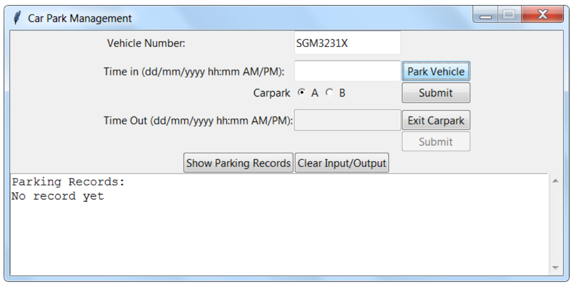


Figure Q4(a)



```
Parking Records:
No record yet
```

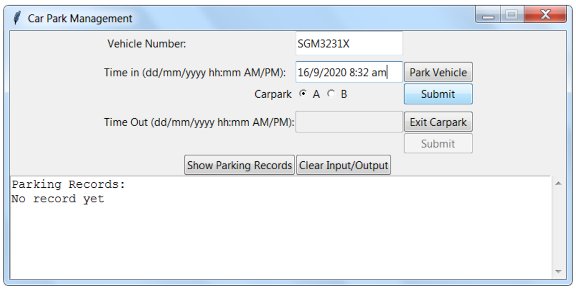
Figure Q4(b)



Vehicle Number: SGM3231X

```
Parking Records:
No record yet
```

Figure Q4(c)

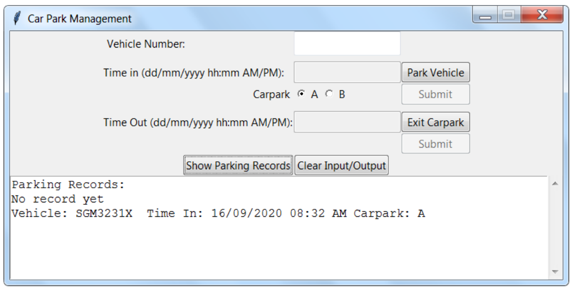


Vehicle Number: SGM3231X

Time in (dd/mm/yyyy hh:mm AM/PM): 16/9/2020 8:32 am

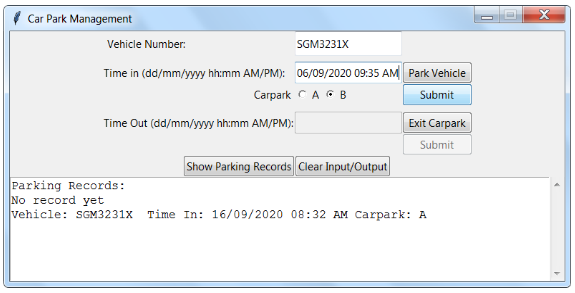
```
Parking Records:
No record yet
```

Figure Q4(d)



```
Parking Records:
No record yet
Vehicle: SGM3231X  Time In: 16/09/2020 08:32 AM Carpark: A
```

Figure Q4(e)



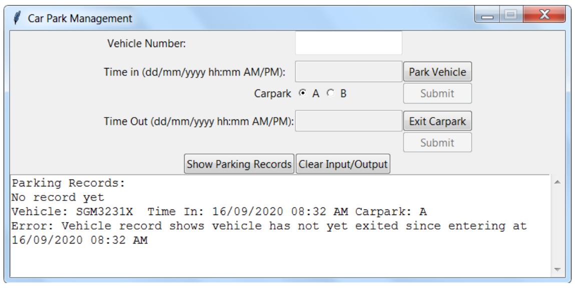
Vehicle Number: SGM3231X

Time in (dd/mm/yyyy hh:mm AM/PM): 06/09/2020 09:35 AM

Carpark ○ A ● B

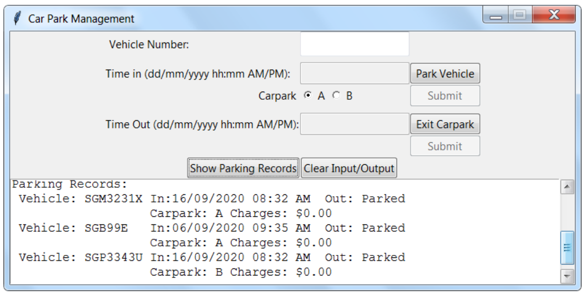
```
Parking Records:
No record yet
Vehicle: SGM3231X  Time In: 16/09/2020 08:32 AM Carpark: A
```

Figure Q4(f)



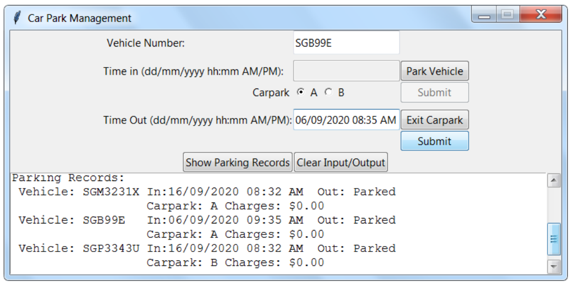
```
Parking Records:
No record yet
Vehicle: SGM3231X  Time In: 16/09/2020 08:32 AM Carpark: A
Error: Vehicle record shows vehicle has not yet exited since entering at
16/09/2020 08:32 AM
```

Figure Q4(g)



```
Parking Records:
 Vehicle: SGM3231X In:16/09/2020 08:32 AM  Out: Parked
                   Carpark: A Charges: $0.00
 Vehicle: SGB99E   In:06/09/2020 09:35 AM  Out: Parked
                   Carpark: A Charges: $0.00
 Vehicle: SGP3343U In:16/09/2020 08:32 AM  Out: Parked
                   Carpark: B Charges: $0.00
```

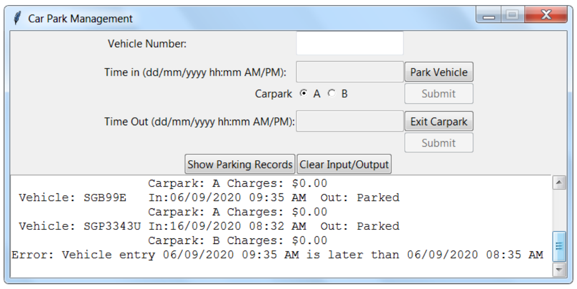
Figure Q4(h)



Vehicle Number: SGB99E

Time Out (dd/mm/yyyy hh:mm AM/PM): 06/09/2020 08:35 AM

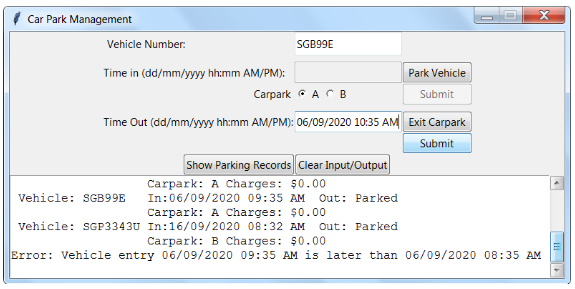
```
Parking Records:
 Vehicle: SGM3231X In:16/09/2020 08:32 AM  Out: Parked
                   Carpark: A Charges: $0.00
 Vehicle: SGB99E   In:06/09/2020 09:35 AM  Out: Parked
                   Carpark: A Charges: $0.00
 Vehicle: SGP3343U In:16/09/2020 08:32 AM  Out: Parked
                   Carpark: B Charges: $0.00
```



```
                   Carpark: A Charges: $0.00
 Vehicle: SGB99E   In:06/09/2020 09:35 AM  Out: Parked
                   Carpark: A Charges: $0.00
 Vehicle: SGP3343U In:16/09/2020 08:32 AM  Out: Parked
                   Carpark: B Charges: $0.00
Error: Vehicle entry 06/09/2020 09:35 AM is later than 06/09/2020 08:35 AM
```

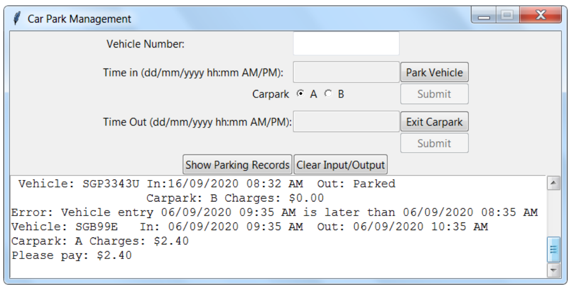
Figure Q4(i)

Figure Q4(j)



Vehicle Number: SGB99E
Time in (dd/mm/yyyy hh:mm AM/PM):
Carpark [✓ A] B
Time Out (dd/mm/yyyy hh:mm AM/PM): 06/09/2020 10:35 AM

```
                   Carpark: A Charges: $0.00
 Vehicle: SGB99E   In:06/09/2020 09:35 AM  Out: Parked
                   Carpark: A Charges: $0.00
 Vehicle: SGP3343U In:16/09/2020 08:32 AM  Out: Parked
                   Carpark: B Charges: $0.00
Error: Vehicle entry 06/09/2020 09:35 AM is later than 06/09/2020 08:35 AM
```

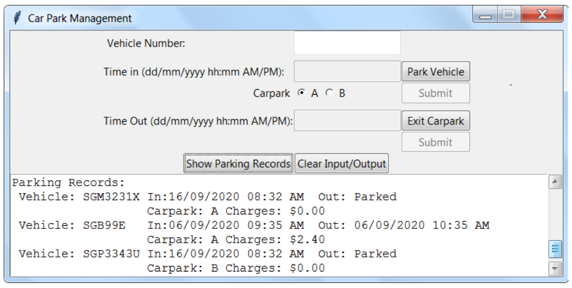


Vehicle Number:
Time in (dd/mm/yyyy hh:mm AM/PM):
Carpark [✓ A] B
Time Out (dd/mm/yyyy hh:mm AM/PM):

```
 Vehicle: SGP3343U In:16/09/2020 08:32 AM  Out: Parked
                   Carpark: B Charges: $0.00
Error: Vehicle entry 06/09/2020 09:35 AM is later than 06/09/2020 08:35 AM
Vehicle: SGB99E   In: 06/09/2020 09:35 AM  Out: 06/09/2020 10:35 AM
Carpark: A Charges: $2.40
Please pay: $2.40
```

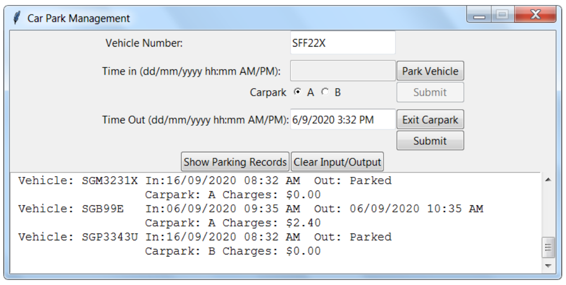
Figure Q4(k)

Figure Q4(l)



Vehicle Number:
Time in (dd/mm/yyyy hh:mm AM/PM):
Carpark [✓ A] B
Time Out (dd/mm/yyyy hh:mm AM/PM):

```
Parking Records:
 Vehicle: SGM3231X In:16/09/2020 08:32 AM  Out: Parked
                   Carpark: A Charges: $0.00
 Vehicle: SGB99E   In:06/09/2020 09:35 AM  Out: 06/09/2020 10:35 AM
                   Carpark: A Charges: $2.40
 Vehicle: SGP3343U In:16/09/2020 08:32 AM  Out: Parked
                   Carpark: B Charges: $0.00
```

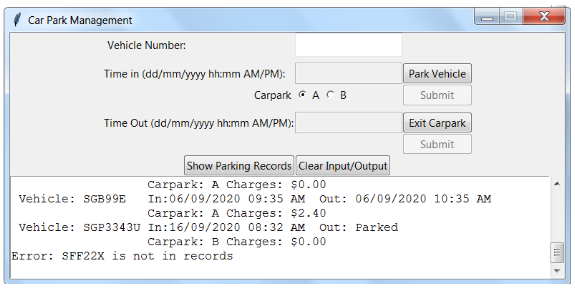


Vehicle Number: SFF22X
Time in (dd/mm/yyyy hh:mm AM/PM):
Carpark [✓ A] B
Time Out (dd/mm/yyyy hh:mm AM/PM): 6/9/2020 3:32 PM

```
 Vehicle: SGM3231X In:16/09/2020 08:32 AM  Out: Parked
                   Carpark: A Charges: $0.00
 Vehicle: SGB99E   In:06/09/2020 09:35 AM  Out: 06/09/2020 10:35 AM
                   Carpark: A Charges: $2.40
 Vehicle: SGP3343U In:16/09/2020 08:32 AM  Out: Parked
                   Carpark: B Charges: $0.00
```

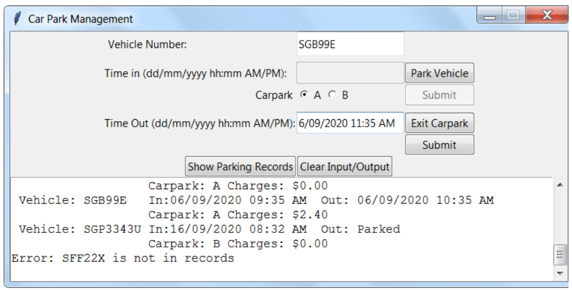
Figure Q4(m)

Figure Q4(n)



Vehicle Number:
Time in (dd/mm/yyyy hh:mm AM/PM):
Carpark [✓ A] B
Time Out (dd/mm/yyyy hh:mm AM/PM):

```
                   Carpark: A Charges: $0.00
 Vehicle: SGB99E   In:06/09/2020 09:35 AM  Out: 06/09/2020 10:35 AM
                   Carpark: A Charges: $2.40
 Vehicle: SGP3343U In:16/09/2020 08:32 AM  Out: Parked
                   Carpark: B Charges: $0.00
Error: SFF22X is not in records
```

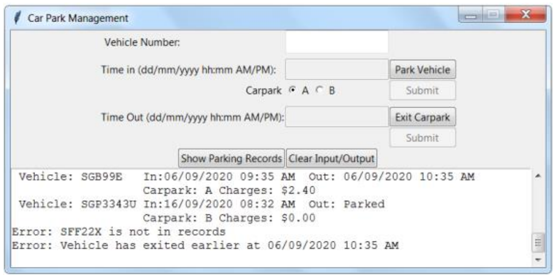


Vehicle Number: SGB99E
Time in (dd/mm/yyyy hh:mm AM/PM):
Carpark [✓ A] B
Time Out (dd/mm/yyyy hh:mm AM/PM): 6/09/2020 11:35 AM

```
                   Carpark: A Charges: $0.00
 Vehicle: SGB99E   In:06/09/2020 09:35 AM  Out: 06/09/2020 10:35 AM
                   Carpark: A Charges: $2.40
 Vehicle: SGP3343U In:16/09/2020 08:32 AM  Out: Parked
                   Carpark: B Charges: $0.00
Error: SFF22X is not in records
```

Figure Q4(o)

Figure Q4(p)



Vehicle Number:
Time in (dd/mm/yyyy hh:mm AM/PM):
Carpark [✓ A] B
Time Out (dd/mm/yyyy hh:mm AM/PM):

```
 Vehicle: SGB99E   In:06/09/2020 09:35 AM  Out: 06/09/2020 10:35 AM
                   Carpark: A Charges: $2.40
 Vehicle: SGP3343U In:16/09/2020 08:32 AM  Out: Parked
                   Carpark: B Charges: $0.00
Error: SFF22X is not in records
Error: Vehicle has exited earlier at 06/09/2020 10:35 AM
```

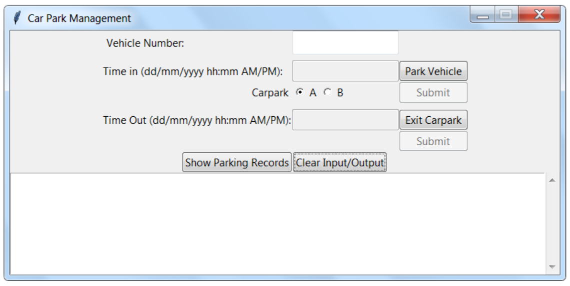


Figure Q4(q)

Figure Q4(r)

- Show Parking Records
  Figure Q4(b), Q4(h) and Q4(m) show the result of clicking on the button Show Parking Records. The content of the dictionary is displayed in the scrolltext. Figure Q4(b) shows the case when the dictionary is empty. Figure Q4(h) and Figure Q4(m) show the case when dictionary is non-empty.

- Park Vehicle
  Figure Q4(c) shows the result of clicking on the button Park Vehicle. Both the textfields for Time In and the Submit button under Park Vehicle are enabled. A user can enter data into the textfield Time In and click on the Submit button, as shown in Figure Q4(d). Note also that the textfield for Time Out is cleared, and the textfield for Time Out and the Submit button under Exit Carpark are disabled.

- Submit button under Park Vehicle
  Figure Q4(d) shows that the Submit button has become clickable. There are three cases:
  - The vehicle has an entry in the dictionary but no time out has been recorded.
    This means that the vehicle has not exited and so, is an error case. Thus, we should not update the dictionary entry. An error message is displayed in the scrolltext. Refer to Figure Q4(f) and Figure Q4(g).
  - The vehicle has an entry in the dictionary and a time out has been recorded.
    The entry is reused to record the new parking. The time in and carpark are recorded, time out set to None, and parking charges set to 0. The parking detail is displayed in the scrolltext as shown in Figure Q4(e).
  - The vehicle does have not an entry in the dictionary yet.
    A new parking entry is created for the vehicle. The time in and carpark are recorded, time out set to None and parking charges set to 0. The parking detail is displayed on the scrolltext as shown in Figure Q4(e).

  In all three cases, the three textfields are cleared and both Submit buttons disabled.

- Exit Carpark
  Figure Q4(i) and Figure Q4(k) show the result of clicking on the button Exit Carpark. The textfield for Time Out and the Submit button under Exit Carpark have become enabled. Figure Q4(i) and Figure Q4(k) also show that data has been entered into the textfield Time In. Note that the textfield for Time Out has been cleared, and the textfield for Time Out and the Submit button under Exit Carpark are disabled.

- Submit button under Exit Carpark
  Figure Q4(j), Figure Q4(l), Figure Q4(o) and Figure Q4(q) show the result of clicking the Submit button. There are four cases:
  - The vehicle has an entry in the dictionary and the user input for a time out is earlier than the recorded time in.
    This case is depicted in Figure Q4(i) which results in Figure Q4(j). An error message is displayed in the scrolltext as shown in Figure Q4(j).
  - The vehicle has an entry in the dictionary and a time out has not been recorded.
    This case is depicted in Figure Q4(k) which results in Figure Q4(l). The entry is updated to reflect the time out and the parking charge incurred.
  - The vehicle does not have an entry in the dictionary yet
    This case is depicted in Figure Q4(n) which results in Figure Q4(o). An error message is displayed in the scrolltext as shown in Figure Q4(o).

- The vehicle has an entry in the dictionary and a time out has already been recorded.
  This case is depicted in Figure Q4(p) which results in Figure Q4(q). An error message is displayed in the scrolltext as shown in Figure Q4(q).

In all four cases, the textfields are cleared, the textfields Time In and Time Out and both Submit button are disabled.

- Clear Input/Output

Refer to Figure Q4(r). The GUI reverts to the one shown in Figure Q4(a), with dictionary is still intact.

You do not need to implement the event handling for this button.

(13 marks)

---- END OF ASSIGNMENT ----