NATIONAL SENIOR CERTIFICATE EXAMINATION
NOVEMBER 2023

INFORMATION TECHNOLOGY: PAPER I

Time: 3 hours 150 marks

PLEASE READ THE FOLLOWING INSTRUCTIONS CAREFULLY

1. This question paper consists of 18 pages. Please check that your question paper is complete.

2. This question paper is to be answered using object-oriented programming principles. Your program must make sensible use of methods and parameters.

3. This paper is divided into two sections. All candidates must answer both sections.

4. This paper is set in programming terms that are not specific to any particular programming language (Java/Delphi) or database (Access/MySQL/JavaDB).

5. Make sure that you answer the questions in the manner described because marks will be awarded for your solution according to the specifications that are given in the question.

6. Only answer what is asked in each question. For example, if the question does not ask for data validation, then no marks are awarded for it, and therefore no code needs to be written for data validation.

7. If you cannot get a section of code to work, comment it out so that it will not be executed and so that you can continue with the examination. If possible, try to explain the error to aid the marker.

8. When accessing files from within your code, DO NOT use full path names for the files, as this will create problems when the program is marked on a computer other than the one you are writing on. Merely refer to the files using their names and extensions, where necessary.

9. Your programs must be coded in such a way that they will work with any data and not just the sample data supplied or any data extracts that appear in the question paper. You are advised to look at the supplied data files carefully.

10. Make sure that routines such as searches, sorts and selections for arrays are developed from first principles, and that you do not use the built-in features of a programming language for any of these routines.

11. All data structures must be defined and declared by you, the programmer. You may not use components provided within the interface to store and later retrieve data.

12. Read the whole question paper before you choose a data structure. You may find that there could be an alternative method of representing the data that will be more efficient considering the questions that are asked in the paper.

13. You must save all your work regularly on the disk you have been given, or the disk space allocated to you for this examination. You should also create a backup of the original files before you start in case the original version is accidentally modified by your solution.

14. If your examination is interrupted by a technical problem such as a power failure, you will, when you resume writing, be given only the time that was remaining when the interruption began, to complete your examination. No extra time will be given to catch up on work that was not saved.

15. Make sure that your examination number appears as a comment in every program that you code as well as on every page of hard copy that you hand in.

16. Print a code listing of all the programs/classes that you code. Printing must be done after the examination. You will be given half an hour to print after the examination is finished. Your teacher will tell you what arrangements have been made for the printing of your work.

17. You should be provided with the following two folders (in bold) and files. These files are to be used as data for this examination. Note that the database files are provided in MS Access, JavaDB and MySQL format. Ensure that you are able to open the files with the packages that you will use to code your solutions to this examination.

    Section A:
    RentMyParking.mdb
    RentMyParking_JavaDB.sql
    RentMyParking_MySQL.sql
    SQLAnswerSheet.rtf
    SQLBrowser.exe

    Section B:
    parkings.txt

SCENARIO

RentMyParking is a company that provides parking bays to car owners. You have been hired to help manage their database system. The database stores the parking bay details and the related vehicle details. Parking bay owners rent their parking bay at a daily rate set by the owner. Premium parking bays are covered and include security. A registered vehicle can be parked at any of the bays and one bay can house any of the vehicles provided it is on a different day.

You are provided with a database called **RentMyParking**.

SECTION A STRUCTURED QUERY LANGUAGE

QUESTION 1

tblCars describes the vehicle's details.

FIELDS	DATA TYPE	DESCRIPTION
CarRegistration	TEXT	The car registration number, used as the primary key of the table.
Model	TEXT	The model of the car.
Colour	TEXT	The colour of the car.
Owner	TEXT	The owner of the car.

The first 10 records in tblCars

CarRegistration	Model	Colour	Owner
BBC 123 MP	Volkswagen Golf	Red	Veli Williams
BCY214 FS	Mazda CX-5	Red	Percy Hlanti
BJW 595 NC	Land Rover RangeRover	White	Keagan Makhurubetshi
CA 691-963	Subaru Outback	Black	Karabo Dolly
CA 741-789	BMW 3 Series	White	Innocent Coetzee
CDF 671 GP	Ford Mustang	Blue	Bradley Jordaan
CH 23 BF GP	Honda Civic	Red	Gabriela Tau
CW 478-321	Tesla ModelS	Silver	Bongeka Mothwa
DGE 962 NC	Mercedes-Benz C-Class	Black	Sifiso Nthite
DSM 032 NW	Audi A4	White	Ronwen Makhabane

tblParkings contains the parking bay details.

FIELDS	DATA TYPE	DESCRIPTION
ParkingID	INTEGER	A unique autonumber identification for each parking bay. This is the primary key.
Address	TEXT	The physical address of the parking bay.
DailyRate	DOUBLE	The daily rate to rent the parking bay.
DateAdded	DATE	The date the parking was added to the database.
PremiumParking	BOOLEAN	A boolean field indicating if the parking is covered and secure.

The first 10 records of tblParkings

ParkingID	Address	DailyRate	DateAdded	PremiumParking
1	262 Mockingbird Junction	102.50	2021-06-12	Yes
2	439 Daystar Alley	132.50	2018-12-09	No
3	12 Heffernan Center	77.50	2020-07-10	No
4	51 Doe Crossing Street	132.50	2019-07-18	No
5	6 Fisk Lane	75.00	2020-10-12	No
6	82 Mosinee Court	77.50	2020-06-20	No
7	33234 Waubesa Court	125.00	2020-01-30	Yes
8	9015 Ohio Place	110.00	2020-01-20	No
9	2 Little Fleur Way	137.50	2017-12-02	No
10	654 Thackeray Plaza	97.50	2018-11-26	No

tblRentedParkings contains the dates a parking bay was rented with the vehicle's registration.

FIELDS	DATA TYPE	DESCRIPTION
CarRegistration	TEXT	The car registration number. A foreign key to tblCars.
ParkingID	INTEGER	The parking bay number. A foreign key to tblParkings.
StartDate	DATE	The start date for the parking reservation.
EndDate	DATE	The end date for the parking reservation.

The first 10 records of tblRentedParkings

tblRentedParkings			
CarRegistration	ParkingID	StartDate	EndDate
BBC 123 MP	5	2023/06/10	2023/06/13
BBC 123 MP	12	2023/01/04	2023/01/10
BCY214 FS	13	2023/05/03	2023/05/17
BCY214 FS	15	2023/05/06	2023/05/19
BCY214 FS	16	2023/02/14	2023/03/03
BJW 595 NC	15	2023/01/30	2023/02/06
BJW 595 NC	24	2023/04/23	2023/04/28
CA 691-963	1	2023/09/11	2023/09/26
CA 691-963	13	2023/04/26	2023/05/01
CA 741-789	18	2023/04/12	2023/04/26

1.1 Display the bays that are premium parking bays sorted in ascending order of the daily rate. List all the fields in the table.

The correct output is shown below:

ParkingID	Address	DailyRate	DateAdded	PremiumParking
23	15002 Gina Pass	70	2017/12/20	Yes
19	41333 Northland Place	82	2019/12/30	Yes
1	262 Mockingbird Junction	102.5	2021/06/12	Yes
14	8415 Sherman Crossing	115	2022/01/22	Yes
7	33234 Waubesa Court	125	2020/01/30	Yes
11	54 Jenifer Terrace	128	2021/02/19	Yes
18	176 Hansons Center	150	2018/03/18	Yes

(4)

1.2 Display the cars with the newer Gauteng registration numbers. A newer Gauteng registration number ends with 'GP' and has a space for the third character. For example, CH 23 BF GP is a new Gauteng registration, but CDF 671 GP is not. List all the fields in the table.

The correct output is shown below:

CarRegistration	Model	Colour	Owner
CH 23 BF GP	Honda Civic	Red	Gabriela Tau
JM 18 HT GP	GMC Sierra	Silver	Karabo Singh

(5)

1.3 The manager at **RentMyParking** is concerned that some normal (non-premium) bays added in 2022 may be more expensive than premium bays. She wants to see a list of all the premium bay's rental costs as well as any normal bay joined in 2022 charging more than R135.00.

List the **Address**, **DailyRate**, **DateAdded** and **PremiumParking** fields for parking bays that are either premium parking or have a daily rate above R135.00 joined in 2022.

The correct output is shown below:

Address	DailyRate	DateAdded	PremiumParking
262 Mockingbird Junction	102.5	2021/06/12	Yes
33234 Waubesa Court	125	2020/01/30	Yes
54 Jenifer Terrace	128	2021/02/19	Yes
8415 Sherman Crossing	115	2022/01/22	Yes
176 Hansons Center	150	2018/03/18	Yes
41333 Northland Place	82	2019/12/30	Yes
15002 Gina Pass	70	2017/12/20	Yes
15695 Vernon Drive	150	2022/02/13	No

(6)

1.4 Display the details of parking bays with the highest daily rate. List all the fields in the table.

The correct output is shown below:

ParkingID	Address	DailyRate	DateAdded	PremiumParking
18	176 Hansons Center	150.00	2018-03-18	Yes
29	15695 Vernon Drive	150.00	2022-02-13	No

(5)

1.5 Display the cars that have not been rented. These cars will not have their registration number listed in the **tblRentedParkings** table. Display the **CarRegistration** and **Model** fields.

The correct output is shown below:

CarRegistration	Model
CDF 671 GP	Ford Mustang
CH 23 BF GP	Honda Civic
JM 18 HT GP	GMC Sierra
ND 47856	Porsche 911
RBG 746 GP	Toyota Camry
RDW 112 GP	Volkswagen Beetle

(4)

1.6 Determine how many times each parking bay has been rented out. Show the **Address** and the number of times rented for those parking bays that have been rented 3 or more times.

The correct output is shown below:
Note the row order may vary.

Address	TimesRented
176 Hansons Center	3
535 Goodland Plaza	3
6796 Straubel Trail	3

(7)

1.7 Calculate the rental amount whenever a vehicle has used a parking bay. The rental amount is the difference in days between the **StartDate** and **EndDate** multiplied by the **DailyRate**. Display the **CarRegistration**, **Owner**, **DailyRate**, **StartDate**, **EndDate** and the rental amount as a field called **RentAmount**.

The first 10 records of the output are shown below:

CarRegistration	Owner	DailyRate	StartDate	EndDate	RentAmount
BBC 123 MP	Veli Williams	75	2023/06/10	2023/06/13	225
BBC 123 MP	Veli Williams	87.5	2023/01/04	2023/01/10	525
BCY214 FS	Percy Hlanti	118	2023/05/06	2023/05/19	1534
BCY214 FS	Percy Hlanti	77.5	2023/05/03	2023/05/17	1085
BCY214 FS	Percy Hlanti	145	2023/02/14	2023/03/03	2465
BJW 595 NC	Keagan Makhurubetshi	135	2023/04/23	2023/04/28	675
BJW 595 NC	Keagan Makhurubetshi	118	2023/01/30	2023/02/06	826
CA 691-963	Karabo Dolly	102.5	2023/09/11	2023/09/26	1537.5
CA 691-963	Karabo Dolly	77.5	2023/04/26	2023/05/01	387.5
CA 741-789	Innocent Coetzee	150	2023/04/12	2023/04/26	2100

(9)

1.8 Change the **Model** field in **tblCars** to store 'VW' instead of 'Volkswagen'. Create a SQL statement to edit the **Model** field by changing all cars whose **Model** starts with 'Volkswagen' to start with 'VW'.

*Below is a list of cars whose **Model** fields are changed to 'VW':*

CarRegistration	Model	Updated Model
BBC 123 MP	Volkswagen Golf	VW Golf
RDW 112 GP	Volkswagen Beetle	VW Beetle

(6)

1.9 Write a query that will remove all the parking bays added before 2018.

Below is the list of deleted parking bays:

ParkingID	Address	DailyRate	DateAdded	PremiumParking
9	2 Little Fleur Way	137.5	2017/12/02	No
12	5 Southridge Alley	87.5	2017/09/23	No
15	6796 Straubel Trail	118	2017/09/19	No
23	15002 Gina Pass	70	2017/12/20	Yes

(4)

50 marks

SECTION B OBJECT ORIENTATED PROGRAMMING

SCENARIO:

RentMyParking is testing a new paperless parking ticket system using RFID tags to record each vehicle's entry and exit times. A vehicle may enter and leave the parking garage multiple times, and at the end of the month, the owner is invoiced for the times they used the parking garage.

The parking garage offers daily and long-term parking. A vehicle with long-term parking is allocated to a permanent parking bay with a number. A vehicle can park in this bay for several hours or days.

For daily parking, the vehicle must exit the parking garage before midnight on the same day they entered. If the vehicle stays overnight, the owner will be fined R850.00.

The following rates are used for daily and long-term parking.

Parking Type	**Rate**
Daily	R24.50 per hour, R850.00 fine for overstay
Long term	R250.00 per day

Sample test data, stored in the text file called **parkings.txt**, has been generated for vehicles with daily and long-term parking over a month.

Below are the first 10 lines of the text file:

```
GEY 268 FS;2023/09/01 10:51;2023/09/01 12:01
RBG 746 GP;2023/09/03 05:57;2023/09/04 13:03;C3
BBC 123 MP;2023/09/03 11:18;2023/09/04 16:11
CA 691-963;2023/09/04 05:40;2023/09/04 12:43;D4
CHZ 231 L;2023/09/04 06:12;2023/09/04 17:15
FBC 852 MP;2023/09/04 07:36;2023/09/06 16:39;B9
RBG 746 GP;2023/09/05 05:55;2023/09/11 06:16;C3
BBC 123 MP;2023/09/05 09:30;2023/09/05 14:45
CA 691-963;2023/09/06 06:56;2023/09/07 05:06;D4
BBC 123 MP;2023/09/07 09:17;2023/09/08 01:10
```

Each line in the text file represents a vehicle's daily or long-term parking entry and exit into the parking garage.

Daily

The details for a vehicle's daily parking entry and exit are stored in the text file as follows:
<registration number>;<entry date and time>;<exit date and time>

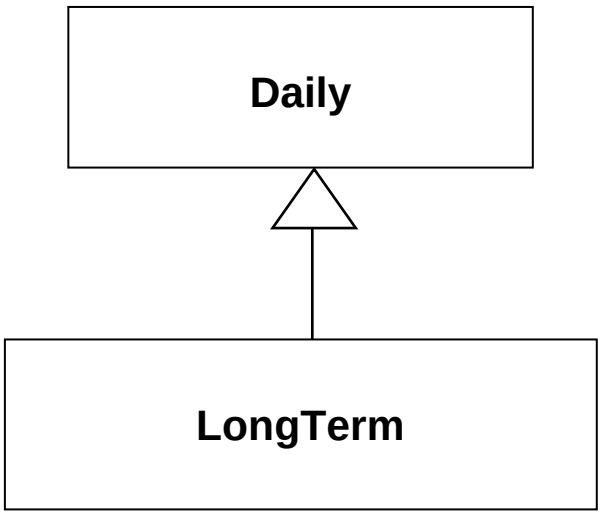
- **Registration number:** a string storing the vehicle's registration number linked to the parking tag.
- **Entry:** the entry date and time in the format yyyy/MM/dd HH:mm.
- **Exit:** the exit date and time in the format yyyy/MM/dd HH:mm.

LongTerm

The details of a long-term parking entry and exit are stored in the text file as follows:
<registration number>;<entry date and time>;<exit date and time>;<parking bay>

- **Registration number:** a string storing the vehicle's registration number linked to the parking tag.
- **Entry:** the entry date and time in the format yyyy/MM/dd HH:mm.
- **Exit:** the exit date and time in the format yyyy/MM/dd HH:mm.
- **Parking bay:** a string storing the assigned parking bay for this vehicle.

You will need to create two classes with the parent **Daily** class and the child **LongTerm** class, as shown in the diagram below:



QUESTION 2

Use the class diagram below to create the **Daily** class. Use this class to create objects to store the daily parking entry details. The diagram indicates the required fields and methods.

Daily
Fields: – registrationNumber : string – entryDateTime : DateTime – exitDateTime : DateTime + HOURLYRATE = 24.5 : real + FINEAMOUNT = 850.0 : real
Methods: + Constructor(inRN : string, inEY : DateTime , inET : DateTime) + getRegistrationNumber() : string + getEntryDateTime() : DateTime + getExitDateTime() : DateTime + getParkingFee() : real + toString() : string

2.1 Create a new class called **Daily** with **registrationNumber**, **entryDateTime** and **exitDateTime** fields as indicated above. Use an appropriate date-time object to store the **entryDateTime** and **exitDateTime** fields. (4)

2.2 Add two **constant static**/**class** fields, **HOURLYRATE** and **FINEAMOUNT,** as shown in the diagram. (2)

2.3 Code a constructor method for the class that will accept the string **inRN** for the **registrationNumber** field, a date time named **inEY** representing the **entryDateTime** field and a date time named **inET** representing the **exitDateTime** field. Use these parameters to assign values to the associated fields. (3)

2.4 Create accessor/getter methods for the **registrationNumber**, **entryDateTime** and **exitDateTime** fields of the class. (2)

2.5 Create the method **getParkingFee** to determine the amount to be charged for each daily entry and exit to the parking garage.

The method must:

- determine if the entry and exit date are the same, and if so,
  - calculate the number of hours the vehicle has been in the garage;
  - multiply the number of hours by the **HOURLYRATE** field;
- if the entry and exit day are not the same, the parking fee will be the value stored in the **FINEAMOUNT** field. You may assume that a vehicle is never more than one day overdue.

For example:

Vehicle registration number: `BJW 595 NC`
Entry date and time: `2023-09-13 06:29`
Exit date and time: `2023-09-13 21:03`
has been in the parking garage for 15 hours
the parking fee will be 15 × R24.50 = R367.50 (8)

2.6 Add a **toString** method to the class that will return a **string** containing **registrationNumber**, **entryDateTime**, **exitDateTime** and **parkingFee**. The **entryDateTime** and **exitDateTime** must appear on a new line.

`"Registration:∇"`**`registrationNumber`**`<tab>"Fee:∇R"`**`parkingFee`**
`"Entry:∇"`**`entryDateTime`**`" Exit:∇"`**`exitDateTime`**

Note that a ∇ indicates a single space.

Example 1:
```
Registration: GEY 268 FS       Fee: R122.5
Entry: 2023-09-17T12:21 Exit:2023-09-17T18:03
```

Example 2 - If there was a fee due to overstaying:
```
Registration: BBC 123 MP        Fee: R850.0
Entry: 2023-09-03T11:18 Exit:2023-09-04T16:11
```

(4)
[23]

QUESTION 3

Use the class diagram below to create a class called **LongTerm**. This class will inherit from the **Daily** class and will be used to create objects to store long-term parking entries. The diagram below indicates the required fields and methods.

LongTerm
Fields: – parkingBay : string + RATEPERDAY = 250.0 : real
Methods: + Constructor(inRN : string, inEY : datetime , inET : datetime , inPB : string) + getParkingBay() : string + getParkingFee() : real + toString() : string

3.1 Create a new class called **LongTerm** that extends the **Daily** class. (2)

3.2 Add the **parkingBay** field as indicated by the diagram. (1)

3.3 Add a **constant static/class** field called **RATEPERDAY**, as shown in the diagram. (1)

3.4 Code a constructor method to initialise the **registrationNumber**, **entryDateTime** and **exitDateTime** fields of the **Daily** parent class and the field **parkingBay** of the **LongTerm** child class. (4)

3.5 Create an **accessor** method for the **parkingBay** field. (1)

3.6 Code a method to override the **getParkingFee** method in the parent class. The method should determine the number of days of the long-term parking entry multiplied by the rate per day field. Return the result as a real.

Note there will be no charge for vehicles that park in a long-term bay for less than one day, as the number of days will be 0.

Example 1:

Vehicle registration number: FBC 852 MP
Entry date and time: 2023-09-09 11:52
Exit date and time: 2023-09-13 16:43

has been in the long-term parking for 4 days.
The parking fee is 4 × R250.00 = R1 000.00.

Example 2:

Vehicle registration number: `RBG 746 GP`
Entry date and time: `2023-09-11 10:39`
Exit date and time: `2023-09-11 20:37`

has been in the long-term parking for less than a day. The parking fee is R0.00. (5)

3.7 Code a **toString** method that will override the parent class's **toString** method, add the **parkingBay** field to the parent's **toString** on a new line and return a string in the following format:

```
"Registration:∇"registrationNumber<tab>"Fee:∇R"parkingFee
"Entry:∇"entryDateTime" Exit:∇"exitDateTime
"Parking Bay:∇"parkingBay
```

Note that a ∇ indicates a single space.

Example:
```
Registration: CA 691-963         Fee: R1250.0
Entry: 2023-09-24T07:14 Exit: 2023-09-29T08:04
Parking Bay: D4
```

(3)
[17]

QUESTION 4

4.1 Create a class called **ParkingManager**. (1)

4.2 Add two instance variables to this class:
- An **array** called **pArr** to store 50 **Daily** or **LongTerm** objects.
- An integer called **size** to record the number of objects added to the array.

These two variables should not be accessible outside the class. (4)

4.3 Code a constructor method for the class that will read the contents of the file **parkings.txt**. Each line of the file contains entries and exits of vehicles using daily or long-term parking.

Your method should do the following:
- Read each line from the file and extract the data
- Instantiate either a **Daily** or **LongTerm** object based on the line's extracted data
- Add the **Daily** or **LongTerm** object to the next available position in the array
- Increase the **size** (11)

4.4 Code a **toString** method to combine the values of each object in the array **pArr** into a string. The details of each daily or long-term object should be separated by a blank line. Use the object's **toString** methods you created in Questions 2.6 and 3.7. (4)

4.5 Create a method named **sortByRegistration** to sort the objects in the array called **pArr** in alphabetical order of the registration numbers. (6)
[26]

QUESTION 5

5.1 Create a **ParkingUI** class with a simple text-based user interface and output. (1)

5.2 Instantiate a **ParkingManager** object named **pm** in an appropriate place in the class. (2)

5.3 **Sort** and display the daily and long-term parking entries in alphabetical order of the registration number.
The output of the first and last six array objects are shown below:
Note that each registration number's entry and exit order may differ depending on the sort algorithm used in Question 4.5.

```
Registration: ABC 123 GP          Fee: R500.0
Entry: 2023-09-25T08:10 Exit: 2023-09-27T14:22
Parking Bay: B2

Registration: BBC 123 MP          Fee: R122.5
Entry: 2023-09-05T09:30 Exit: 2023-09-05T14:45

Registration: BBC 123 MP          Fee: R850.0
Entry: 2023-09-07T09:17 Exit: 2023-09-08T01:10

Registration: BBC 123 MP          Fee: R147.0
Entry: 2023-09-14T09:26 Exit: 2023-09-14T15:26

Registration: BBC 123 MP          Fee: R850.0
Entry: 2023-09-03T11:18 Exit: 2023-09-04T16:11

Registration: BJW 595 NC          Fee: R73.5
Entry: 2023-09-09T09:58 Exit: 2023-09-09T13:20

......
Registration: GEY 268 FS          Fee: R122.5
Entry: 2023-09-17T12:21 Exit: 2023-09-17T18:03

Registration: RBG 746 GP          Fee: R0.0
Entry: 2023-09-11T10:39 Exit: 2023-09-11T20:37
Parking Bay: C3

Registration: RBG 746 GP          Fee: R250.0
Entry: 2023-09-14T14:27 Exit: 2023-09-15T14:40
Parking Bay: C3

Registration: RBG 746 GP          Fee: R250.0
Entry: 2023-09-03T05:57 Exit: 2023-09-04T13:03
Parking Bay: C3

Registration: RBG 746 GP          Fee: R250.0
Entry: 2023-09-17T06:39 Exit: 2023-09-18T05:12
Parking Bay: C3

Registration: RBG 746 GP          Fee: R1500.0
Entry: 2023-09-05T05:55 Exit: 2023-09-11T06:16
Parking Bay: C3
```

(2)
[5]

QUESTION 6

RentMyParking needs to create invoices for each vehicle owner who has used the parking garage. Since each vehicle has used the garage multiple times in the month, the parking fee for each entry and exit must be calculated and added to a monthly total.

6.1 Since each vehicle registration can have multiple array objects, a unique sequence of vehicle registration numbers must be created. Code a method called **getRegistrationList** to create an alphabetical string of the registration numbers stored in the array called **pArr**. The string must not contain any duplicates. Separate each registration number with the # character.

The method should return the following string:

```
ABC 123 GP#BBC 123 MP#BJW 595 NC#CA 691-963#CHZ 231 L#FBC
852 MP#GEY 268 FS#RBG 746 GP#
```

(6)

6.2 Code a method called **generateBilling** to return a string with invoices for each vehicle registration number. (1)

The method must do the following:

6.2.1 Call the **getRegistrationList** method and assign it to a string called **cars**. (1)

6.2.2 Use the string called **cars** to do the following:

- use the '#' character to isolate the registration numbers. (3)
- for each registration number extracted:
  - search the array **pArr** for the matching registration number and determine the total parking fee using the methods you coded in Questions 2.5 and 3.6. (5)
  - if the registration number has a long-term parking bay, record the bay number. (4)
  - append (join) each vehicle's registration number, followed by the month's parking entries and exits with the total amount due for the month to the invoice string. (3)
  - include the bay number at the end of the invoice string if the vehicle has a long-term parking bay. (2)
  - for daily parking with a monthly total above R1500.00, include the message " Consider renting long-term parking". (2)

6.3 In the **ParkingUI** class, call the **generateBilling** method to display the invoice amounts for each vehicle registration number.

Your output should appear as follows:
Note the order of entries and exits below each registration number may differ depending on the type of sort used in Question 4.5.

```
ABC 123 GP
2023-09-25T08:10 2023-09-27T14:22 500.0
Total parking fees due: R500.0
Long-term bay number: B2

BBC 123 MP
2023-09-05T09:30 2023-09-05T14:45 122.5
2023-09-07T09:17 2023-09-08T01:10 850.0
2023-09-14T09:26 2023-09-14T15:26 147.0
2023-09-03T11:18 2023-09-04T16:11 850.0
Total parking fees due: R1969.5
Consider renting long-term parking

BJW 595 NC
2023-09-09T09:58 2023-09-09T13:20 73.5
2023-09-16T07:49 2023-09-16T12:20 98.0
2023-09-21T11:53 2023-09-22T12:16 850.0
2023-09-13T06:29 2023-09-13T21:03 343.0
Total parking fees due: R1364.5

CA 691-963
2023-09-08T07:35 2023-09-09T03:49 250.0
2023-09-06T06:56 2023-09-07T05:06 250.0
2023-09-11T13:07 2023-09-11T19:43 0.0
2023-09-24T07:14 2023-09-29T08:04 1250.0
2023-09-04T05:40 2023-09-04T12:43 0.0
Total parking fees due: R1750.0
Long-term bay number: D4

CHZ 231 L
2023-09-20T16:43 2023-09-20T17:33 0.0
2023-09-04T06:12 2023-09-04T17:15 269.5
2023-09-07T10:05 2023-09-08T10:25 850.0
2023-09-18T11:01 2023-09-18T11:53 0.0
Total parking fees due: R1119.5

FBC 852 MP
2023-09-09T11:52 2023-09-13T16:43 1000.0
2023-09-13T08:37 2023-09-14T05:53 250.0
2023-09-04T07:36 2023-09-06T16:39 500.0
Total parking fees due: R1750.0
Long-term bay number: B9
```



```
GEY 268 FS
2023-09-01T10:51 2023-09-01T12:01 24.5
2023-09-07T12:05 2023-09-07T12:42 0.0
2023-09-11T15:00 2023-09-11T21:01 147.0
2023-09-13T05:27 2023-09-13T16:22 245.0
2023-09-17T12:21 2023-09-17T18:03 122.5
Total parking fees due: R539.0

RBG 746 GP
2023-09-11T10:39 2023-09-11T20:37 0.0
2023-09-14T14:27 2023-09-15T14:40 250.0
2023-09-03T05:57 2023-09-04T13:03 250.0
2023-09-17T06:39 2023-09-18T05:12 250.0
2023-09-05T05:55 2023-09-11T06:16 1500.0
Total parking fees due: R2250.0
Long-term bay number: C3
```

(2)
[29]

100 marks

Total: 150 marks