INTERNATIONAL SECONDARY CERTIFICATE EXAMINATION
NOVEMBER 2022

COMPUTER SCIENCE: PAPER I

Time: 3 hours 150 marks

PLEASE READ THE FOLLOWING INSTRUCTIONS CAREFULLY

1. This question paper consists of 19 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. Ensure 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/output files 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:
    GroupHoldings_Access.mdb
    GroupHoldings_JavaDB.sql
    GroupHoldings_MySQL.sql
    SQLAnswerSheet.rtf
    SQLBrowser.exe

    Section B:
    orders.txt

SCENARIO:

The Restaurant Group is a company that manages restaurants. Two restaurants serve only halaal food and the other four serve food whether it is halaal or not. The company employs managers, chefs, and servers to oversee the day-to-day operations of the restaurant. There is a variety of dishes that are offered for sale at the restaurants. These are referred to as menu items. Chefs employed by the company have received training in how to prepare some of the menu items. Multiple chefs might be trained to prepare any given menu item. Once trained, a chef is given a rating that indicates how skilled they are in preparing that specific item. The possible ratings are 'Average', 'Good' and 'Excellent'.

SECTION A SQL

QUESTION 1

All the data on restaurants owned by The Restaurant Group, the staff employed, the menu items and the chef training are stored in a database called **GroupHoldings**.

tblRestaurants contains the details of all the restaurants owned by The Restaurant Group.

FIELDS	DATA TYPE	DESCRIPTION
RestID	INTEGER	A unique auto-numbered identification number for each restaurant.
RestName	TEXT	The name of the restaurant.
Halaal	BOOLEAN	Whether the food served by the restaurant is entirely halaal.

All the records in tblRestaurants:

RestID	RestName	Halaal
1	Royal Rolls	FALSE
2	Selma's Eatery	TRUE
3	Delicious Dishes	FALSE
4	Prime Meals	TRUE
5	Smooth	FALSE
6	Traditional Foods	FALSE

tblStaff contains the details of the staff employed by The Restaurant Group.

FIELDS	DATA TYPE	DESCRIPTION
StaffID	INTEGER	A unique auto-numbered identification number for each staff member.
FirstName	TEXT	The staff member's first name.
Surname	TEXT	The staff member's surname.
JobTitle	TEXT	The job title or role of the staff member.
FullTime	BOOLEAN	Whether or not the staff member works full-time.
EmploymentDate	DATE	The date the staff member was employed.
RestID	INTEGER	Which restaurant the staff member works at. This is a foreign key to tblRestaurants.

The first 10 records of tblStaff:

tblStaff table						
StaffID	**FirstName**	**Surname**	**JobTitle**	**FullTime**	**EmploymentDate**	**RestID**
1	Olivia	Spencer	Manager	TRUE	2003/03/05	1
2	Gugulethu	Gcuma	Server	TRUE	2008/11/27	1
3	Shirley	Renwick	Server	TRUE	1996/08/05	2
4	Bindi	Anthony	Manager	TRUE	2016/09/15	3
5	Kuhle	Mbongwe	Manager	TRUE	2004/07/27	2
6	Rafiq	Varma	Server	TRUE	2000/04/15	3
7	Thobeka	Sibeko	Chef	FALSE	2015/11/14	5
8	Zanele	Mathibela	Server	TRUE	1994/01/27	4
9	Laura	Olivier	Chef	FALSE	2014/08/15	4
10	Kagiso	Ntshangase	Chef	TRUE	2005/03/05	3

tblMenuItems contains the details of all menu items offered by The Restaurant Group

FIELDS	DATA TYPE	DESCRIPTION
MenuItemID	INTEGER	A unique auto-numbered identification number for each menu item.
MenuName	TEXT	The name of the item on the menu.
Cost	DOUBLE	The cost to prepare the menu item.
Halaal	BOOLEAN	Whether the menu item is halaal.

The first 10 records of tblMenuItems:

tblMenuItems table			
MenuItemID	**MenuName**	**Cost**	**Halaal**
1	Steak Roll	51,50	FALSE
2	Veggie Roll	37,75	TRUE
3	Chips	15,00	TRUE
4	Battered Hake	66,25	TRUE
5	Grilled Salmon	52,00	TRUE
6	Onion Rings	21,75	TRUE
7	Baked Potato	32,50	TRUE
8	Breadsticks	15,25	TRUE
9	Chicken Pot Pie	65,25	TRUE
10	Spaghetti Bolognaise	70,00	FALSE

tblTraining contains the details of the menu items the chefs have been trained to prepare.

FIELDS	DATA TYPE	DESCRIPTION
StaffID	INTEGER	The identification number of the staff member. This is a foreign key to tblStaff.
MenuItemID	INTEGER	The identification number of the menu item. This is a foreign key to tblMenuItems.
SkillLevel	TEXT	The level of expertise the staff member has at making the menu item.

The first 10 records of tblTraining:

tblTraining table		
StaffID	MenuItemID	SkillLevel
7	3	Good
7	4	Good
7	5	Average
7	16	Good
7	17	Average
7	23	Excellent
7	25	Average
7	26	Good
7	27	Average
7	38	Excellent

1.1 Display the details of the staff whose job title is 'Server'. Sort the output according to the date of their employment with the employees who have been employed the longest first. Sample output is given below.

Note only the first six records are shown.

StaffID	FirstName	Surname	JobTitle	FullTime	EmploymentDate	RestID
12	Fezile	Gambu	Server	FALSE	1991/02/23	4
35	Phillip	Glass	Server	FALSE	1992/11/03	2
14	Yibanathi	Khanyeza	Server	TRUE	1993/11/07	5
8	Zanele	Mathibela	Server	TRUE	1994/01/27	4
16	Liu	Yang	Server	TRUE	1996/06/19	6
3	Shirley	Renwick	Server	TRUE	1996/08/05	2

(4)

1.2 Every year staff members receive a bonus on the anniversary of their employment date. For example, if someone were employed in November, they would receive a bonus every year in November. Display the **FirstName**, **Surname** and **EmploymentDate** of all staff members who were employed in the current month. Include a random integer between 50 and 100, inclusive, which is the bonus they will receive. Sample output is given below:

Note the records shown will vary depending on the current month. The bonus amount will vary according to the random number generated.

FirstName	Surname	EmploymentDate	Bonus
Zhang	Wei	2002/10/18	55
Janine	Grill	1996/10/23	97

(6)

1.3 Determine the average cost to prepare the menu items that have Soup anywhere in their menu name. Name this field **SoupCost**. The average cost does not need to be rounded off. The correct output is shown below:

SoupCost
70.2916666

(5)

1.4 The Restaurant Group wants to ensure that all part-time staff who work at a halaal restaurant receive appropriate training. Display the **RestName**, **FirstName**, **Surname** and **JobTitle** of all staff employed at a halaal restaurant who are part-time employees. The correct output is shown below:

RestName	FirstName	Surname	JobTitle
Prime Meals	Laura	Olivier	Chef
Prime Meals	Fezile	Gambu	Server
Prime Meals	Lesedi	Even	Chef
Selma's Eatery	Todd	Pencilben	Manager
Prime Meals	Zhang	Wei	Manager
Selma's Eatery	Phillip	Glass	Server

(6)

1.5 The Restaurant Group needs a report on which restaurants are understaffed for each job title. For each restaurant, display the **RestName**, **JobTitle** and the number of employees with that job title. Only display rows where there is only a single staff member working in each role. The correct output is shown below:

RestName	JobTitle	NumStaff
Prime Meals	Manager	1
Smooth	Manager	1
Traditional Foods	Chef	1
Traditional Foods	Manager	1

(7)

1.6 The Restaurant Group needs to know which dishes their staff members can cook with a skill level of 'Excellent'. Display a list of the staff member's first name, surname and the menu name of all menu items that can be cooked with a skill level of 'Excellent'. The list should be sorted by menu name and then surname. Sample output is given below:

Note only the first five records are shown.

FirstName	Surname	MenuName
Nontasasa	Jali	Beef Ribs
Lesedi	Even	Breadsticks
Nontasasa	Jali	Buffalo Wings
Laura	Olivier	Buffalo Wings
Nontasasa	Jali	Chicken Pesto

(7)

1.7 Some chicken menu items were named using the Italian word for 'chicken', 'pollo'. The Restaurant Group wants these renamed to 'chicken' instead of 'pollo'. Write a query to change all menu items that start with 'Pollo' to start with 'Chicken' instead. For example, 'Pollo Pizza' should be renamed to 'Chicken Pizza'. (6)

1.8 The Restaurant Group is introducing some vegan steaks and ribs. For each menu item ending with steak or ribs in the name, insert a matching record in **tblMenuItems**. The menu name must be the existing menu name preceded by 'Vegan'. The cost will be $10 less than the equivalent non-vegan menu item. All the vegan menu items will be halaal. The inserted records are shown below:

MenuItemID	MenuName	Cost	Halaal
48	Vegan Rump Steak	90.25	TRUE
49	Vegan Pork Ribs	85.75	TRUE
50	Vegan Beef Ribs	95.25	TRUE
51	Vegan Prime Rib Steak	100.75	TRUE
52	Vegan T-Bone Steak	85.00	TRUE

(9)

50 marks

SECTION B OBJECT-ORIENTATED PROGRAMMING

The restaurant, Royal Rolls, sells pre-cooked meals to customers so that they don't need to cook after work. Customers can collect their own order of food or choose a delivery time for their meal to be delivered to their door. The restaurant sells only steak rolls, vegetarian rolls and portions of chips as pre-cooked meals. All orders are placed the day before they are due. When an order is received by the restaurant it is stored in a text file called **orders.txt**.

Royal Rolls delivers to five suburbs: Nodale, Heresford, Everyvale, Allmare and Thereville. The suburbs are divided into delivery zones depending on distance from the restaurant. Nodale and Heresford are in delivery zone A. Everyvale and Allmare are in delivery zone B. Thereville is in delivery zone C.

- Orders to delivery zone A take approximately 5 minutes to deliver and have a delivery fee of $5.00.
- Orders to delivery zone B take approximately 10 minutes to deliver and have a delivery fee of $7.50.
- Orders to delivery zone C take approximately 15 minutes to deliver and have a delivery fee of $10.00

Royal Rolls employs four delivery drivers. Any driver can deliver to any zone but can only handle a maximum of five orders per night. The latest time that a delivery driver will leave to deliver an order is 22:00.

The management at Royal Rolls needs your help to create a program that will manage the received orders and create a delivery list for their drivers.

Order Class

This class will represent the basic details of all orders being prepared for collection by customers:

- **orderID** – the order code used to identify the order.
- **numSteakRolls** – the number of steak rolls the customer wants.
- **numVeggieRolls** – the number of vegetarian rolls the customer wants.
- **numChips** – the number of portions of chips the customer wants.
- **instructions** – the special instructions for the kitchen, such as 'Extra garnish' or 'No sauce'. If there are no special instructions this field will store 'None'.

The details of the collection orders are stored in a text file called **orders.txt.** A line of the file representing the data of an individual order has the following format:

```
<orderID> ; <numSteakRolls> ; <numVeggieRolls> ; <numChips> ;
<instructions>
```

DeliveryOrder Class

This class will represent the basic details of all orders and additional details related to the delivery. It is the child class of the **Order** class. The **DeliveryOrder** class has the following additional fields.

- **address** – the address the order must be delivered to.
- **deliveryTime** – the time the order is meant to be delivered.
- **deliveryZone** – the delivery zone that the address is categorised into.

The details of the delivery orders are also stored in the **orders.txt** text file. A line of the file representing the data of an individual order has the following format:

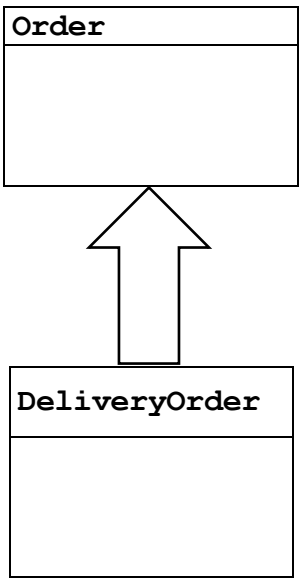
<**orderID**> ; <**numSteakRolls**> ; <**numVeggieRolls**> ; <**numChips**> ; <**instructions**> ; <**address**> ; <**deliveryTime**>

Note that **deliveryZone** is not stored in the text file.

The first ten lines of the text file, containing data about **Order**s and **DeliveryOrder**s are below:

```
K0349;5;4;1;None
V4403;0;4;2;None;57 Garam Masala Street Nodale;20:15:00
A7155;1;4;2;None
D8286;4;0;0;None
P7068;1;4;4;Extra salt;73 Garlic Powder Road Allmare;18:30:00
J5085;1;1;1;None
J7812;2;1;2;None;49 Salt Street Everyvale;19:25:00
I9108;1;4;5;None;32 Sumbala Avenue Nodale;20:15:00
I7109;3;2;4;None
R2226;4;2;6;None
```

A UML diagram showing the relationship between the **Order** class and the **DeliveryOrder** class is given below. The fields and methods of the **Order** class and the **DeliveryOrder** class are given in the UML diagrams in Question 2 and Question 3.



QUESTION 2

Use the class diagram below to create a new class called **Order**. This class will be used to store the details of an order.

Order
Fields: - orderID : string - numSteakRolls : integer - numVeggieRolls : integer - numChips : integer - instructions : string
Methods: + Constructor(inOID : string, inNumSR : integer, inNumVR : integer, inNumC : integer, inInstr : string) + getOrderID() : string + getCost() : real + toString() : string

2.1 Create a new class named **Order** with the **orderID**, **numSteakRolls**, **numVeggieRolls**, **numChips** and **instructions** fields as shown in the class diagram. These fields must not be accessible outside the class. (3)

2.2 Add a parameterised constructor method to the class that will assign the values to the **orderID**, **numSteakRolls**, **numVeggieRolls**, **numChips** and **instructions** fields of the class. (4)

2.3 Add an accessor method for the **orderID** field of the class. (2)

2.4 Code a **getCost** method to return a real representing the total food cost of the order. A steak roll costs \$51.50, a vegetarian roll costs \$37.75 and a portion of chips costs \$15.00. Use the formula below to calculate the total food cost:

```
numSteakRolls × 51.5 + numVeggieRolls × 37.75 + numChips
× 15
```
(3)

2.5 Code a **toString** method to return a string combining all the fields in the following format:

```
'Order:'<space>orderID<newline>
'Steak Rolls:'<space>numSteakRolls<newline>
'Veggie Rolls:'<space>numVeggieRolls<newline>
'Chips:'<space>numChips<newline>
'Instructions:'<space>instructions<newline>
'Total:'<space>'$'[cost]
```

For example:

```
Order: K0349
Steak Rolls: 5
Veggie Rolls: 4
Chips: 1
Instructions: None
Total: $423.5
```

(4)
[16]

QUESTION 3

Use the class diagram below to create a new class called **DeliveryOrder** that inherits from the **Order** class. This class will be used to store the details of an order that needs to be sent for delivery. The diagram below indicates the field and methods that are required.

DeliveryOrder
Fields: - address: string - deliveryTime : Time - deliveryZone : character + LATESTDEPARTURE = 22:00:00 : Time
Methods: + Constructor(inOID : string, inNumSR : integer, inNumVR : integer, inNumC : integer, inInst : string, inAddr : string, inTime : Time) + getAddress() : string + getDeliveryZone() : character + getDeliveryTime() : Time + getCost() : real + getLatestDeparture() : Time + toString() : string

3.1 Create a new class named **DeliveryOrder** with the **address**, **deliveryTime** and **deliveryZone** fields as shown in the class diagram. **DeliveryOrder** must inherit from the **Order** class. These fields must not be accessible outside the class. (4)

3.2 Write code to add the constant **LATESTDEPARTURE** to the class and assign it the value 22:00:00. This field must be accessible outside the class. (2)

3.3 Write code to create a parameterised constructor method for the class that will accept parameters for the **orderID**, **numSteakRolls**, **numVeggieRolls**, **numChips**, **instructions, address**, and **orderTime**.
The **deliveryZone** field must be initialised depending on the **address** field.
- If the **address** contains the word 'Nodale' or 'Heresford' then **deliveryZone** must be assigned A.
- If the **address** contains the word 'Everyvale' or 'Allmare' then **deliveryZone** must be assigned B.
- All other **address**es must be assigned C. (6)

3.4 Create accessor methods for the **address**, **deliveryTime** and **deliveryZone** fields. (2)

3.5 Create code to override the **getCost** method from the **Order** class. This new method must return the total cost of the food including the delivery charge.
- Deliveries to zone A cost $5.00.
- Deliveries to zone B cost $7.50.
- Deliveries to zone C cost $10.00. (5)

3.6 Create a **toString** method that will return a string that combines the values of the fields in the **DeliveryOrder** class, including the inherited fields from the **Order** class. The method uses the **toString** method of the **Order** class. The format should be as follows:

```
'Order:'<space>orderID<newline>
'Steak Rolls:'<space>numSteakRolls<newline>
'Veggie Rolls:'<space>numVeggieRolls<newline>
'Chips:'<space>numChips<newline>
'Instructions:'<space>instructions<newline>
'Total:'<space>$[cost]<newline>
'Delivery to zone'<space>deliveryZone<space>'included.'
```

For example:

```
Order: V4403
Steak Rolls: 0
Veggie Rolls: 4
Chips: 2
Instructions: None
Total: $186.0
Delivery to zone A included.
```

(4)

3.7 Create a **getLatestDeparture** method that will determine the latest time a driver can leave with the order and still deliver the food before the **deliveryTime**.

- Deliveries to zone A will take 5 minutes to deliver.
- Deliveries to zone B will take 10 minutes to deliver.
- Deliveries to zone C will take 15 minutes to deliver.

(5)

[28]

QUESTION 4

4.1 Create a class named **OrderManager**. (1)

4.2 Create two fields for the class. The first must be an array called **oArr** to store 100 **Order** and **DeliveryOrder** objects. The second must be a counter called **size** to keep track of the number of objects added to the array called **oArr**. The fields should not be accessible outside the class. (4)

4.3 Create a constructor method that will read all the contents of the text file named **orders.txt** containing the information for collection and delivery orders. Each line read from the file will add either one **Order** object or one **DeliveryOrder** object to the array called **oArr**. The method should do the following:

- Check if the file **orders.txt** exists.
- Display an error message if the file does not exist.
- Open the file for reading.
- Loop through all the lines of the text file. In each iteration of the loop:
  - Read the line and split the data into the separate parts.
  - Create either an **Order** object or a **DeliveryOrder** object depending on the data on a line. Store the object in the next available position in the array called **oArr**.
- Update the counter variable called **size**. (11)

4.4 Write code to create a **toString** method. This method should return a string that contains details about all orders, whether they are delivered or collected. Each order's details should be separated by a blank line. Use the **toString** methods created in the previous classes. (5)

4.5 Write code to create a method named **getTotalTakeoutCost**. This method should return a real that contains the total cost of all the orders for delivery. Orders that are not being delivered should not be included in the total. (6)

[27]

QUESTION 5

5.1 Write code to create a text-based user interface called **OrderUI** that will allow simple input and output. (1)

5.2 Declare and instantiate an **OrderManager** object in an appropriate place in the code. (1)

5.3 Write code to call the appropriate method in the **OrderManager** class to display a list of all the **Order** and **DeliveryOrder** objects. Sample output of the first two and last two orders are shown below:

```
Order: K0349
Steak Rolls: 5
Veggie Rolls: 4
Chips: 1
Instructions: None
Total: $423.5

Order: V4403
Steak Rolls: 0
Veggie Rolls: 4
Chips: 2
Instructions: None
Total: $186.0
Delivery to zone A included.
…

Order: F4047
Steak Rolls: 2
Veggie Rolls: 3
Chips: 0
Instructions: No sauce
Total: $221.25
Delivery to zone A included.

Order: G2918
Steak Rolls: 4
Veggie Rolls: 5
Chips: 0
Instructions: None
Total: $402.25
Delivery to zone B included.
```

(1)

5.4 Write code to call the appropriate method in the **OrderManager** class to display the total cost of all orders that are being delivered. The output is given below:

```
Total cost of delivery orders: $6970.0
```

(1)
[4]

QUESTION 6

6.1 Code a helper method called **validateTime** that accepts a **DeliveryOrder** as a parameter. If the departure time for the **DeliveryOrder** is later than the latest departure time defined in **Question 3.2** then the method should return null. Otherwise, the method should return the departure time for the driver in the following format:

'Driver must leave by'<space>departureTime

For example:
Driver must leave by 20:10

[6]

QUESTION 7

7.1 Code a method named **deliveryLists** that will create a list of deliveries for each driver. The restaurant employs four drivers, and they can be referred to as Driver 1, Driver 2, Driver 3 and Driver 4. Each driver can deliver a maximum of five orders but may deliver fewer than five orders. The method should allocate the delivery orders to the drivers using the following guidelines

- If a delivery order requires a driver to depart later than the latest departure time defined in **Question 3.2**, it should be rejected.
- The first delivery order should be allocated to Driver 1, the second should be allocated to Driver 2, the third should be allocated to Driver 3 and the fourth should be allocated to Driver 4.
- This repeats for all subsequent delivery orders so, for example, the fifth order should be allocated to Driver 1, the sixth to Driver 2 and so on.
- Once all drivers each have five delivery orders allocated to them, any further delivery orders should be rejected.

The method should return a string containing the delivery orders.

Each driver's deliveries should have a heading in the following format:

```
'Driver'<space>driverNumber<space>'deliveries'<newline>
```

Underneath the heading each of the driver's deliveries should be displayed in the following format:

```
'OrderID'<newline>
'Deliver to:'<space>address<newline>
'Driver must leave by'<space>departureTime<newline>
<blank line>
```

After the four drivers' deliveries have been displayed, the rejected deliveries should be displayed using the following format:

```
'Rejected deliveries'<newline>
OrderID
```

Sample output is provided in Question 7.2. (17)

7.2 Call the **deliveryLists** method in the appropriate place and display each driver's delivery lists. The correct output is displayed below:

```
Driver 1 deliveries
V4403
Deliver to: 57 Garam Masala Street Nodale
Driver must leave by 20:10

O6528
Deliver to: 7 Nutmeg Avenue Everyvale
Driver must leave by 18:20

Y9187
Deliver to: 73 Peppercorn Street Nodale
Driver must leave by 21:35

F2499
Deliver to: 49 Banga Road Nodale
Driver must leave by 20:05

B7586
Deliver to: 35 Dukkah Street Thereville
Driver must leave by 21:30

Driver 2 deliveries
P7068
Deliver to: 73 Garlic Powder Road Allmare
Driver must leave by 18:20

C9400
Deliver to: 4 Iru Street Everyvale
Driver must leave by 20:35

C0662
Deliver to: 67 Banga Road Nodale
Driver must leave by 20:50

H7682
Deliver to: 73 Lemon Zest Street Thereville
Driver must leave by 19:20

R7891
Deliver to: 17 Marigold Avenue Heresford
Driver must leave by 19:25

Driver 3 deliveries
J7812
Deliver to: 49 Salt Street Everyvale
Driver must leave by 19:10

A8437
Deliver to: 8 Banga Road Nodale
Driver must leave by 19:45
```

```
T8009
Deliver to: 91 Fennel Crescent Allmare
Driver must leave by 17:59

K5399
Deliver to: 20 Garam Masala Street Nodale
Driver must leave by 19:05

X0068
Deliver to: 74 Garlic Powder Road Allmare
Driver must leave by 21:40

Driver 4 deliveries
I9108
Deliver to: 32 Sumbala Avenue Nodale
Driver must leave by 20:10

S1265
Deliver to: 30 Sumbala Avenue Nodale
Driver must leave by 21:10

M3777
Deliver to: 41 Pepper Road Heresford
Driver must leave by 18:05

O9936
Deliver to: 16 Cinnamon Avenue Thereville
Driver must leave by 17:50

F4047
Deliver to: 6 Blue Basil Avenue Heresford
Driver must leave by 21:00

Rejected deliveries
D4850
G2918
```

(2)
[19]

100 marks

Total: 150 marks