INTERNATIONAL SECONDARY CERTIFICATE EXAMINATION
NOVEMBER 2024

COMPUTER SCIENCE: 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. 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:
    IceCreams.mdb
    IceCreams_JavaDB.sql
    IceCreams_MySQL.sql
    SQLAnswerSheet.rtf
    SQLBrowser.exe

    Section B:
    IceCreams.txt

SCENARIO:

The company **Tastee IceCream** sells a variety of ice cream products in normal or mini sizes to stores across the country. **Tastee IceCream** orders its ice cream from various suppliers and has rated the supplier's quality of service. Each supplier order has one or more ice cream products. The quantity of ice creams and delivery details are recorded in each order. The supplier's ice cream product price can change from one order to the next.

SECTION A STRUCTURED QUERY LANGUAGE

QUESTION 1

The data relating to ice creams, suppliers, and orders is stored in an **IceCream** database with four tables.

tblIceCreams contains the ice cream details of the ice cream ordered from suppliers.

FIELDS	DATA TYPE	DESCRIPTION
IceCreamID	INTEGER	A unique auto-numbered identification number for each ice cream. Primary Key field.
IceCreamName	TEXT	The name of the ice cream.
Description	TEXT	The ice-cream description.
Size	TEXT	The ice-cream size (N – Normal, M – Mini).
MarkUp	DOUBLE	A decimal value to determine the selling price.
Discount	BOOLEAN	Whether a discount is applied to this ice cream.

A sample of the first FIVE records in tblIceCreams is shown below:

IceCreamID	IceCreamName	Description	Size	MarkUp	Discount
1	Magnum Classic	Creamy vanilla ice cream coated in a thick layer of milk chocolate.	N	0.042	Yes
2	Magnum Almond	Creamy vanilla ice cream coated in a layer of milk chocolate and almond pieces.	N	0.026	No
3	Magnum White	Creamy vanilla ice cream coated in a thick layer of white chocolate.	N	0.075	Yes

4	Magnum Double Caramel	Creamy vanilla ice cream with a caramel swirl, coated in milk chocolate.	N	0.05	Yes
5	Magnum Intense Dark	Rich dark chocolate coating a creamy vanilla ice cream.	N	0.094	Yes

tblSuppliers contains the supplier's details.

FIELDS	DATA TYPE	DESCRIPTION
SupplierID	INTEGER	A unique auto-numbered identification number for each supplier. Primary Key field.
SupplierName	TEXT	The supplier's name.
ContactPerson	TEXT	The contact person's name at the supplier.
ContactNumber	TEXT	The phone number of supplier contact person.
Rating	INTEGER	**Tastee IceCream's** internal rating of the supplier (1 – Poor, 2 – Below Average, 3 – Average, 4 – Good, 5 – Excellent).

A sample of the first FIVE records in tblSuppliers are shown below:

SupplierID	SupplierName	ContactPerson	ContactNumber	Rating
1	Ola	Ethan Smith	071 234 5678	4
2	Nestle South Africa	Sophia Johnson	082 345 6789	4
3	Gatti Ice Cream	Noah Williams	060 456 7890	2
4	Dairymaid	Olivia Jones	079 567 8901	2
5	Hubertos	Liam Brown	081 678 9012	3

tblOrders contains the recent orders made by the company.

FIELDS	DATA TYPE	DESCRIPTION
OrderID	INTEGER	A unique auto-numbered identification number for each order.
SupplierID	INTEGER	The identification number of the Supplier. This is a foreign key to tblSuppliers.
OrderDate	DATE	Date the order was made.
EmployeeName	TEXT	The name of the employee that captured the order.

A sample of the first FIVE records of tblOrders are shown below:
*Note that the format of the **OrderDate** field may differ.*

OrderID	SupplierID	OrderDate	EmployeeName
1	4	2024/09/04	Jessica Davis
2	4	2024/09/06	Daniel Miller
3	2	2024/09/06	Andrew Moore
4	10	2024/09/06	Sarah Jones
5	10	2024/09/07	Jessica Davis

tblItemsOrdered contains the ice creams ordered from a supplier. The price, quantity of an ice cream product, delivery date and expiry date are included.

Note that ice cream products are not always ordered from the same supplier. Suppliers may change the ice cream prices due to stock availability, demand and costs.

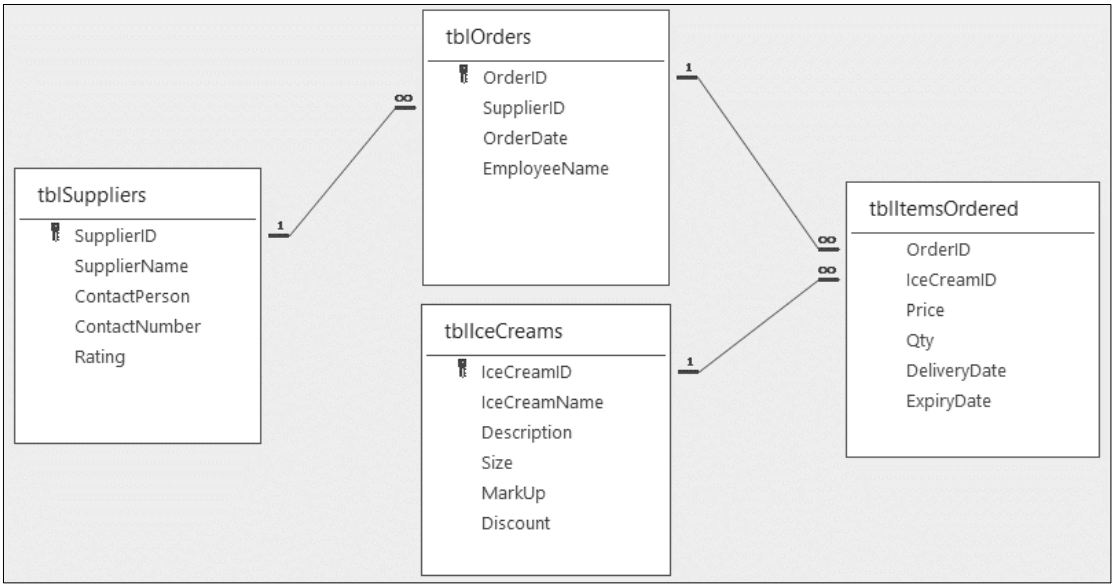
Some orders do not have a **DeliveryDate** or **ExpiryDate** if the order has not been delivered.

FIELDS	DATA TYPE	DESCRIPTION
OrderID	INTEGER	The identification number of the order. This is a foreign key to tblOrders.
IceCreamID	INTEGER	The identification number of the ice cream. This is a foreign key to tblIceCreams.
Price	DOUBLE	The price of each ice cream for the order.
Qty	INTEGER	The number of ice cream items ordered.
DeliveryDate	DATE	The delivery date of the ice cream.
ExpiryDate	DATE	The ice cream's expiry date for the order.

A sample of the first FIVE records of tblItemsOrdered are shown below:

OrderID	IceCreamID	Price	Qty	DeliveryDate	ExpiryDate
1	35	14.25	150	2024/09/13	2024/10/13
1	36	14.55	150	2024/09/14	2024/10/14
2	39	13.15	150	2024/09/20	2024/10/20
2	40	14.95	150	2024/09/14	2024/10/14
3	38	13.11	200	2024/09/14	2024/10/14

Relationship Diagram:



1.1 Display the supplier details who have a rating of above 3.

The correct output is given below.

SupplierID	SupplierName	ContactPerson	ContactNumber	Rating
1	Ola	Ethan Smith	071 234 5678	4
2	Nestle South Africa	Sophia Johnson	082 345 6789	4
10	Boxer Trading	Isabella Taylor	072 123 4567	5

(4)

1.2 Display **IceCreamName**, **Description**, **Size** and **Discount** fields for ice creams with 'berry' in the description. Only display ice creams that can be given a discount.

The correct output is given below.

IceCreamName	Description	Size	Discount
Magnum Ruby	Ruby chocolate coating a creamy ice cream with raspberry swirls.	N	Yes
Ola Rich n Creamy Strawberry	Strawberry-flavoured ice cream coated in a layer of chocolate.	N	Yes
Solero Berry	A mix of raspberry and strawberry sorbet with a creamy vanilla ice cream.	N	Yes
Solero Red Berries	A blend of red berry sorbet with a creamy vanilla ice cream.	N	Yes
Cornetto Strawberry	Strawberry-flavoured ice cream with a strawberry sauce filling, topped with a chocolatey tip.	N	Yes
King Kone Strawberry Cone	Strawberry-flavoured ice cream in a cone with a chocolate coating and nuts.	N	Yes

(5)

1.3 Display the **OrderID**, **IceCreamID**, **Qty** and **Price** for ice creams ordered between the 13th and 20th (inclusive) of September 2024.
Only display details for orders with **odd** numbered **OrderID**.

The correct output is given below.

OrderID	IceCreamID	Qty	Price
1	35	150	14.25
1	36	150	14.55
3	38	200	13.11
3	41	250	12.97
3	45	200	13.21
3	47	200	12.98
5	8	200	10.21
7	53	100	13.98
7	54	100	14.12

(5)

1.4 Display the **EmployeeName** and the number of orders each employee has made. Name this field **OrdersMade**. Only display employees who have ordered from "Boxer Trading" (**SupplierID 10**), in descending order of the number of orders.

The correct output is given below.

EmployeeName	OrdersMade
Jessica Davis	2
Sarah Jones	1
Rachel Wilson	1

(6)

1.5 The company would like to identify ice creams ordered in large quantities.
Display the **IceCreamID** and the total quantity of each ice cream ordered.
Only display ice creams where the total quantity ordered is more than 250.

The correct output is given below.

IceCreamID	Total
7	400
25	330
35	300

(7)

1.6 The company wants to calculate the selling price of ice creams that meet the following criteria:

- a selling price is more than R16.00.
- the ice cream name is 20 characters or less;

Use the following formula to calculate the selling price:

Selling Price = price × (1 + markup)

Display the **IceCreamName**, **OrderID** and the calculated **SellingPrice**.
Ensure that the **SellingPrice** is rounded to TWO decimal places.
Display the result in alphabetical order of **IceCreamName** and then ascending order of **OrderID**.

Ice cream names could be duplicated, as the orders may have different ice cream prices.

The correct output is given below.

IceCreamName	OrderID	SellingPrice
Blueberry Swirl	11	16.79
Blueberry Swirl	13	16.76
Caramel Krunch	11	16.06
Chocolate Swirl	13	17.4
Mint Krunch	11	16.39

(11)

1.7 **Order 20** is incomplete with only one item. Add to **Order 20** all the items ordered for **Order 12**, with a randomly generated quantity (**Qty**) between 125 and 175 (inclusive). Do not provide values for the delivery and expiry date fields.

The complete list of items to be added for **Order 20**, is shown below the first existing order (in italics), with randomly generated quantities (**Qty**).

Note the randomly generated quantities will differ.

OrderID	IceCreamID	Price	Qty	DeliveryDate	ExpiryDate
20	*32*	*14.12*	*100*		
20	4	10.15	158		
20	5	11.35	138		
20	7	10.25	139		

(9)

1.8 Some ice creams have not been ordered from any of the suppliers due to the decrease in popularity. Remove ice creams with the **IceCreamID** of 1, 2, 6 and 10. (3)

50 marks

SECTION B OBJECT-ORIENTATED PROGRAMMING

Section A dealt with **Tastee IceCream's** orders from suppliers. Section B deals with the current stock of ice cream products. In this section, each product's selling price is determined by its category number (1 to 4). The name, category number, markup, qty, and expiry date for all the ice cream products are stored in a text file called **IceCreams.txt**.

IceCream Class
This class will represent the details of ice cream products sold.

- **name** – the ice cream product name
- **category number** – a number from 1 to 4 to determine the selling price
- **markUp** – the markup percentage used with the category to determine the selling price
- **qty** – the quantity of the ice cream product in stock
- **expiry** – the expiry date for this batch of ice cream (assume that the expiry date for all ice creams of the same name will be the same)

Some ice creams contain extra ingredients and possible allergens[1], such as peanut butter.

ExtraIceCream Class
This class will represent the details of ice cream products with extra ingredients and allergens. It is a child class of the **IceCream** class with the following additional fields.

- **extras** – the number of additional ingredients added to the ice cream
- **allergens** – a list of allergen ingredients

Ice creams and extra ice creams are stored in the text file called **IceCreams.txt**.

A line of text representing an ice cream is in the following format:
`<name>;<categoryNum>;<markUp>;<qty>;<expiry>`

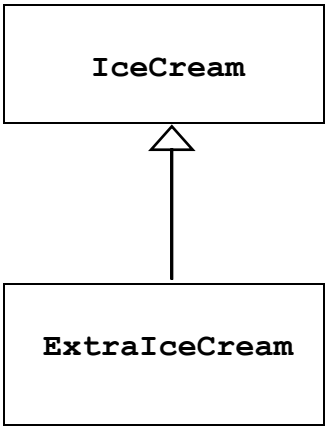
A line of text representing an extra ice cream is in the following format:
`<name>;<categoryNum>;<markUp>;<qty>;<expiry>;<extras>;<allergens>`

The first ten lines of the text file, containing data about **IceCreams** and **ExtraIceCreams** are shown below:

```
Cornetto Caramel;2;12;194;2024-10-03
Cornetto Cookies & Dream;3;13;146;2024-10-13;3;Ground Nuts, Wheat
Cornetto Chocolate;3;13;105;2024-11-05
Cornetto Classico;1;9;180;2024-12-03
Cornetto Peanut Brittle;3;10;166;2024-11-18;2;Peanuts
Cornetto Strawberry;4;10;174;2024-11-14
King Kone Banana Cone;4;12;152;2024-12-11;3;None
King Kone Caramel Cone;3;12;101;2024-12-17;3;Sesame
King Kone Chocolate Cone;2;10;139;2024-10-11
King Kone Coconut Cone;4;12;135;2024-12-01
```

[1] An allergen is a substance that can cause an allergic reaction.

A UML diagram below shows the relationship between the **IceCream** and **ExtraIceCream** classes. The fields and methods of the **IceCream** and **ExtraIceCream** classes 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 **IceCream**. This class will be used to store the details of an ice cream product.

IceCream
Fields: - name : string - categoryNum : integer - markUp : real - qty : integer - expiry : Date
Methods: + Constructor(inName : string, inCategoryNum : integer, inMarkUp : real, inQty : integer, inExpiry : Date) + getName() : string + getExpiry() : Date + getSellingPrice() : real + toString() : string + checkExpiry() : boolean

2.1 Create a new class named **IceCream** with the **name**, **categoryNum**, **markUp**, **qty** and **expiry** fields as shown in the class diagram. These fields must not be accessible outside the class. (3)

2.2 Code the parameterised constructor method for the class that will accept parameters and assign these values to the **name**, **categoryNum**, **markUp**, **qty** and **expiry** fields for the class. (4)

2.3 Code the accessor methods for the **name** and **expiry** fields for the class. (2)

2.4 Code the **getSellingPrice** method to return the ice cream selling price as a real value.

The selling price is calculated using the formula below:

$$\text{Selling Price} = \text{Category Cost} + \frac{\text{Category Cost} \times \text{Markup}}{100}$$

Use the table below to determine the **Category Cost** based on the **categoryNum** of an ice cream product:

CategoryNum	Category Cost
1	10.15
2	11.09
3	11.26
4	11.65

Example: An ice cream with a category number of 1 produces a **Category Cost** of 10.15. If the markup is 12% the selling price is calculated as follows:

$$\text{Selling Price} = 10.15 + \frac{10.15 \times 12}{100}$$

$$\text{Selling Price} = 11.368$$

(5)

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

```
'Ice cream Name:'<space>name<newline>
'Quantity:'<space>qty<newline>
'Expiry Date:'<space>expiry<newline>
'Selling Price:'<space>selling price<newline>
```

For example:

```
Ice cream Name: Cornetto Caramel
Quantity: 194
Expiry Date: 2024-10-03
Selling Price: 12.4208
```

NB. Decimal separator for Selling Price could either be a decimal point (.) or a comma (,). (5)

NB. The **checkExpiry** method will be coded in Question 6.1.

[19]

QUESTION 3

Use the class diagram below to create a new **ExtraIceCream** class that inherits from the **IceCream** class. This class will store the details of an ice cream with extra ingredients or allergens.

The diagram below indicates the field and methods that are required.

ExtraIceCream
Fields: - extras : integer - allergens : string + PRICEPEREXTRA = 12.00 : real
Methods: + Constructor(inName : string, inCategoryNum : integer, inMarkUp : real, inQty : integer, inExpiry : Date inExtras : integer, inAllergens : string) + getAllergens() : string + getSellingPrice() : real + toString() : string

3.1 Create a new class named **ExtraIceCream** with the **extras** and **allergens** fields as shown in the class diagram. Both fields must not be accessible outside the class. The **ExtraIceCream** class must inherit from the **IceCream** class. (4)

3.2 Write code to add the constant **PRICEPEREXTRA** to the class and assign the value 12.00. This field must be accessible outside the class. (2)

3.3 Code the parameterised constructor method for the class that will accept parameters and assign these values to the **name**, **categoryNum**, **markUp**, **qty**, **expiry**, **extras** and **allergens** fields, for the class. (5)

3.4 Code the accessor method for the **allergens** field. (1)

3.5 Write code to override the **getSellingPrice** method from the **IceCream** class. This new method must return the selling price of an ice cream with extra ingredients and allergens as follows:

Selling Price = Selling Price + (Extras × Price Per Extra)

Use the class constant in your calculation.

(4)

3.6 Code the **toString** method that will return a string that combines the values of the fields in the **ExtraIceCream** class, including the inherited fields from the **IceCream** class.
The method must use the **toString** method of the **IceCream** class.

The formatted string is as follows:

```
'Ice cream Name:'<space>name<newline>
'Quantity:'<space>qty<newline>
'Expiry Date:'<space>expiry<newline>
'Selling Price:'<space>selling price<newline>
'Extras:'<space>extras<newline>
'Allergens:'<space>allergens<newline>
```

For example:

```
Ice cream Name: Cornetto Cookies & Dream
Quantity: 146
Expiry Date: 2024-10-13
Selling Price: 48.7238
Extras: 3
Allergens: Ground Nuts, Wheat
```

(4)

[20]

QUESTION 4

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

4.2 Create two fields for the class as follows:

- The first must be an array called **iArr** to store 100 **IceCream** and **ExtraIceCream** objects.
- The second must be a counter called **size** to keep track of the number of objects added to the array.
- The fields should not be accessible outside the class. (4)

4.3 Code a constructor method that will read all the contents of the text file named **IceCreams.txt**, containing the information for ice creams and ice creams with extra ingredients and allergens.

Each line read from the file will be used to add either one **IceCream** object or one **ExtraIceCream** object to the array, **iArr**

The method should do the following:

- Check if the file **IceCreams.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 separate parts.
  - Convert the expiry date into a Date object using the formatting "yyyy-MM-dd".
  - Instantiate either an **IceCream** object or **ExtraIceCream** object, depending on the data on a line.
  - Store the object in the next available position in the array, **iArr**.
  - Update the counter variable called **size**. (13)

4.4 Code a **toString** method. This method must return a string with all the ice cream and extra ice cream details separated by a blank line.
Use the **toString** methods created in the previous classes. (5)

4.5 Code a method named **getAverageExtra**.
This method must return the average price of all the **ExtraIceCream** objects that contain allergens. Round the result to the nearest whole number. (8)

[31]

QUESTION 5

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

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

5.3 Write code to call the appropriate method in the **IceCreamManager** class to display a list of all the **IceCream** and **ExtraIceCream** objects.
A sample output of the first two and last two ice creams are shown below:

```
Ice cream Name: Cornetto Caramel
Quantity: 194
Expiry Date: 2024-10-03
Selling Price: 12.4208

Ice cream Name: Cornetto Cookies & Dream
Quantity: 146
Expiry Date: 2024-10-13
Selling Price: 48.7238
Extras: 3
Allergens: Ground Nuts, Wheat
.
.
.
Ice cream Name: Solero Red Berries
Quantity: 190
Expiry Date: 2024-10-02
Selling Price: 12.9315

Ice cream Name: Solero Watermelon
Quantity: 183
Expiry Date: 2024-11-10
Selling Price: 13.048
```

(1)

5.4 Write code to call the appropriate method in the **IceCreamManager** class to display the average price of all special ice creams that contain allergens.

The output is given below:

```
Extra Ice Creams Allergens Average Price: 43.0
```

(1)

[4]

QUESTION 6

6.1 Code a method called **checkExpiry**, in the **IceCream** class that compares the expiry date of the ice cream with today's date.
Should the expiry date be before today's date, return **true**; otherwise, return **false**. (4)

6.2 Code a method called **expiredIceCreams** in the **IceCreamManager** class to create a text file called **ExpiredIceCreams.txt** with the details of expired ice creams. Use the **toString** methods to format the text written to the file. (8)

6.3 In the **IceCreamUI** class, call the **expiredIceCreams** method in an appropriate place. The **ExpiredIceCreams.txt** should contain the following data, if the code is run today, 18 October 2024:

```
Ice cream Name: Cornetto Caramel
Quantity: 194
Expiry Date: 2024-10-03
Selling Price: 12.4208

Ice cream Name: Cornetto Cookies & Dream
Quantity: 146
Expiry Date: 2024-10-13
Selling Price: 48.7238
Extras: 3
Allergens: Ground Nuts, Wheat

Ice cream Name: King Kone Chocolate Cone
Quantity: 139
Expiry Date: 2024-10-11
Selling Price: 12.199

Ice cream Name: Magnum Intense Dark
Quantity: 154
Expiry Date: 2024-10-17
Selling Price: 12.6112

Ice cream Name: Solero Lime
Quantity: 117
Expiry Date: 2024-10-12
Selling Price: 12.815000000000001

Ice cream Name: Solero Peach
Quantity: 200
Expiry Date: 2024-10-02
Selling Price: 13.048

Ice cream Name: Solero Red Berries
Quantity: 190
Expiry Date: 2024-10-02
Selling Price: 12.9315
```

(1)

[13]

QUESTION 7

7.1 The manufacturer of King Kone ice creams has recalled all their products.

All King Kone ice creams must be removed from the array.

Code a method called **removeIceCreams** in the **IceCreamManager** class that will remove all objects from the array that contain the words **King Kone** in their **name**. The remaining objects in the array, should be one after the other, followed by empty blocks in array. (11)

7.2 In the **IceCreamUI** class, call the **removeIceCreams** method in an appropriate place and then display a list of all the remaining **IceCream** and **ExtraIceCream** objects. (2)

[13]

100 marks

Total: 150 marks