SECTION E DATA AND INFORMATION MANAGEMENT AND SOLUTION DEVELOPMENT

QUESTION 7

The **MobiHealth** database which stores details of the patients who are treated at each mobile clinic, stores the following data in one table called **tblMedicalData:**

Field	Description
PatientID	Unique ID for each patient treated by the clinic
Surname	The surname of the patient
FirstName	The first names of the patient
ConsultID	Unique ID for each consultation
Date	The date of the consultation
Duration	The duration of the consultation
FollowUp	The number of days when the patient must return for another consultation

A patient can have many different consultations. Each consultation will only be with one patient. Each consultation needs a follow up a certain number of days later.

Sample data for **tblMedicalData** is shown below:

PatientID	Surname	FirstName	ConsultID	Date	Duration	FollowUp
1	Bhebhe	Mthunzi	234	01/04/18	20	10
2	Hlongwa	Kagiso	244	02/04/18	25	90
3	Luthuli	Bonginkosi	267	15/10/18	10	30
4	Chonco	Uluthando	315	12/07/18	15	0
5	Khoza	Langa	337	12/07/18	40	90
6	Zindela	Inyoni	219	21/02/18	55	0
7	Luthuli	Mthunzi	288	16/04/18	12	90
4	Chonco	Uluthando	266	17/06/18	18	10
1	Bhebhe	Mthunzi	322	21/03/18	20	90

7.1 It is common practice to normalise database tables.

7.1.1 List THREE reasons why database tables should be normalised.

Reason 1: _____

_____

Reason 2: _____

_____

Reason 3: _____

_____

(3)

7.1.2 Database tables can be normalised to a number of normal forms.

List the characteristics of both the second and third normal forms. (2NF and 3NF)

2NF: _____

_____

_____

(2)

3NF: _____

_____

_____

(2)

7.2 **tblMedicalData** is going to be transformed into TWO new tables: **tblPatients** – for patient specific data and **tblConsultations** – for consultation specific data. In the questions which follow, you must restrict your answers to working with just these two tables, i.e. you may not define any additional tables and you are not required to remove any transitive dependencies.

7.2.1 Define the term *primary key*.

_____

_____

_____

_____

(2)

7.2.2 Define the term *composite key*.

_____

_____

(1)

7.2.3 Write down the best primary key for **tblMedicalData**.

_____

_____

_____

_____

(2)

7.2.4 Complete the following two tables, showing the fields that will be present in the two new tables. Your primary keys must be listed as the first field in each table. Indicate your choice of foreign key below the tables.

tblPatients	**tblConsultations**

Foreign Key: _____

(5)

7.2.5 Define the term *foreign key*.

_____

_____

_____

_____

(2)
[19]

QUESTION 8

A software developer is busy writing an OOP application for **MobiHealth**. One of the classes in the OOP application will be used to define **Patient** objects. There are also classes for **Doctor** objects and **Clinic** objects.

Each **Patient** object will have the following fields:

patientID : integer
surname : string
firstName : string
patientAge : integer
medication : array [20] string
clinicName : string
followUp : integer

A patient is always treated by the same doctor and each doctor only works in one mobile clinic, each of which has a unique name.

All fields of **Patient** objects are private.

8.1 Answer the following question relating to the object fields:

In which class(es) will the fields of each **Patient** object be directly accessible? Tick the box next to the correct option below.

- The Patient, Doctor and Clinic classes ☐
- The Patient and Doctor classes ☐
- Only the Patient class ☐
- The Patient and Clinic classes ☐

(1)

8.2 The **Patient** class will have the following methods:

- A parameterised constructor method which will accept 7 parameters named **p** (integer), **s** (string), **fN** (string), **a** (integer), **m** (string array), **c** (string) and **f** (string);
- Accessor method for the **clinicName** field;
- Mutator method for the **clinicName** field. This method will have a parameter: **cn**, of the correct type to match the type of the corresponding field;
- A *toString* method.

8.2.1 The **Doctor** class has a field which is a Patient object. Both classes have *toString* methods. Is this an example of method overloading, method overriding or neither? Explain your answer.

_____

_____

_____

_____

_____

_____

(3)

8.2.2 You are required to assist the developers by completing a class diagram for the **Patient** class.

Fields
Methods

(9)

8.2.3 Currently the age of a patient is stored as an integer field in the **Patient** object. What would be a better data value and corresponding data type to store in this field which will still allow us to know the patient's age?

_____

(1)

8.3 When the developers are coding the **Patient** class for the application, they will be working with various algorithms. They first represent the algorithms visually by writing them on pieces of paper which they then fit together to make a working algorithm. Some of the pieces of paper have been mixed up and you have been asked to re-assemble one of the algorithms.

The algorithm in question is meant to sort an array of **medication** (a field in the **Patient** class). The developers are using an **improved Bubble Sort** which uses a **flag** to stop the sort if no further swaps are needed.

8.3.1 Included as **Appendix B** is a sheet containing representations of the various pieces of paper that the algorithm was planned on. Each has a letter of the alphabet to identify it. You are required to assemble these into the correct order and write down the correct order, i.e. the letters of the alphabet to get the algorithm to work correctly. NB: Some of the pieces of paper are not relevant to this algorithm. There are exactly the correct number of blocks for a correct answer. You have also been given TWO lines of the sort, G and M. They are in the correct positions.

Sample Answer:

C	G	F	L	D	S	A

Your Answer:

				G				M				

(10)

8.3.2 Does this algorithm sort the array in ascending or descending order?

_____

(1)

8.4 A method to determine the average age of a patient is being coded. The algorithm for the method is included as **Appendix C**.

There are four values in the array: 24 ; 36 ; 18 and 22.

The algorithm will loop through an array of patient ages (named **ageArr**) and calculate the average age. It will also maintain a running average whilst looping through the array. If the running average exceeds 60, an error message will be displayed.

8.4.1 This algorithm has been coded, but is resulting in unexpected output. You are required to complete the following trace table to assist the programmer in finding the error in the algorithm.

line	size	temp	runningAvg	k	count	runningAvg > 60?	DISPLAY
1	4						
2		0					
3			0				
4					0		
4				0			
6		24					
7					1		
8			24				
9						F	
5				1			
6							
7							
8							
9							
5							
6							
7							
8							
9							
10							
5							
6							
7							
8							
9							
10							
11							

(9)

8.4.2 Which line in the algorithm is responsible for the error?

_____

(1)

8.4.3 Is the variable count necessary in this algorithm as it stands? Can it be replaced by k? Justify your answer.

_____

_____

(2)
[37]

Total: 180 marks

ADDITIONAL SPACE (ALL questions)

REMEMBER TO CLEARLY INDICATE AT THE QUESTION THAT YOU USED THE ADDITIONAL SPACE TO ENSURE THAT ALL ANSWERS ARE MARKED.

APPENDIX C

1	size ← 4
2	temp ← 0
3	runningAvg ← 0
4	count ← 0
5	for k ← 0 to size -1
	begin
6	temp ← temp + ageArr[k]
7	count ++
8	runningAvg ← runningAvg + (temp / count)
9	if (runningAvg > 60)
10	display "Error"
	end if
	end loop
11	display (temp / count)