basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

NOVEMBER 2012

MARKS: 120

TIME: 3 hours

This question paper consists of 16 pages and 5 annexures.

INSTRUCTIONS AND INFORMATION

1. The duration of this examination is three hours. Because of the nature of this examination it is important to note that you will not be permitted to leave the examination room before the end of the examination session.

2. With regard to the formulation of the questions in terms of programming, no distinction has been made between the two programming languages in this question paper. Where required, specific instructions have been provided for Delphi and Java candidates respectively.

3. You require the files listed below in order to answer the questions. They are EITHER on a stiffy disk OR CD issued to you OR the invigilator/teacher will tell you where to find them on the hard drive of the workstation you are using OR in a network folder.

Question1_Delphi:	**Question1_Java:**
Question1P.dpr	tblCarnivores.txt
Question1P.res	tblVetVisits.txt
Question1U.dfm	TestQuestion1.java
Question1U.pas	Zoo.java
tblCarnivores.txt	ZooDB.mdb
tblVetVisits.txt	
ZooDB.mdb	

Question2_Delphi:	**Question2_Java:**
DataQ2.txt	DataQ2.txt
Question2P.dpr	Quest2.java
Question2P.res	TestQuestion2.java
Question2U.dfm	
Question2U.pas	
uQuest2.pas	

Question3_Delphi:	**Question3_Java:**
DataQ3_Delphi.txt	DataQ3_Java.txt

If you received the files above on a disk (CD or stiffy), write your examination number on the label.

4. Type in your examination number as a comment in the first line of each program file that contains your programming code.

5. Your program should always be coded to answer the question in the way it has been formulated. You are not allowed to only copy the given output supplied in the question paper.

6. Read ALL the questions carefully. Do not do more than the questions require.

7. To help you to understand each question better, you have to read the entire question before answering any subquestions.

8. Save your work at regular intervals as a precaution against power failures.

9. There might be a technical interruption that prevents you from writing the examination, such as a power failure. When you resume writing the examination, you will be given the time remaining when the interruption began, and an additional 10 minutes.

10. During the examination, you may use the manuals originally supplied with the hardware and software. You may also use the HELP functions of the software. Candidates using Java may use the Java API files. You may NOT use any other resource material.

11. At the end of this examination session, you must hand in the disk or CD with all your work saved on it OR you must make sure that all your work has been saved on the hard drive/network as explained to you by the invigilator/teacher. Ensure that all files can be read.

12. Make printouts of the programming codes of all the programming questions you have done.

13. All printing of programming questions that you have done will take place within an hour of the completion of this examination.

SCENARIO

The Africa Zoo houses a large variety of animals. The animals are kept in different enclosures within specific areas of the zoo. The zoo also has an aquarium and a reptile park.

The zoo requires new software for the administration of the various activities at the zoo.

You are required to complete the following THREE questions using the programming language you have studied.

QUESTION 1: PROGRAMMING AND DATABASE

A Microsoft Office Access database named **ZooDB.mdb**, two text files (**tblCarnivores.txt** and **tblVetVisits.txt**) and an incomplete program are provided in the folder named **Question1_XXXX** where XXXX refers to the programming language you have studied.

The design of the tables in the **ZooDB** database and sample data for each table can be found in **ANNEXURE A**.

Do the following:

- Make a backup copy of the **ZooDB** database **BEFORE** you start answering QUESTION 1. You will need a copy of the original database to be able to test your program thoroughly.

- Rename the given folder for QUESTION 1 by replacing the name of the programming language you have studied with your examination number.

- Open the given incomplete program for QUESTION 1.

- Add your examination number as a comment in the first line of the program file.

- Compile and execute the program. The interface displays eight menu options as indicated in the section labelled **QUESTION 1** in **ANNEXURE B (Delphi)/ANNEXURE C (Java)**.

  NOTE:
  - An error message will be displayed if any of the options A–G are selected, due to the incomplete SQL statements.

  - If you experience any problems using the database or connecting to the database, refer to **ANNEXURE D (Delphi)/ANNEXURE E (Java)** for troubleshooting hints.

  - If you still experience database problems, you must nevertheless do the SQL code and submit it for marking. **Marks will only be awarded for the programming code that contains the SQL statements.**

- Complete the code for each menu option by formulating an appropriate SQL statement to display the results for the respective query as described in QUESTIONS 1.1 to 1.7 below.

  NOTE: The code to some input statements and the code to execute the SQL statements and display the results of the queries have already been written as part of the given code.

1.1 **Menu Option A**

Display all the details of the animals stored in the **tblCarnivores** table, sorted firstly by the **FamilyName** field and secondly by the **ScientificName** field in **alphabetical order**.

Example of the output of the first five records:

EnclosureNo	FamilyName	ScientificName	GeneralName	NumAdults	NumYoung	EnclosureSize	Endangered
ZE7	Canidae	Canis adustus	Side-striped jackal	6	3	46	LE
ZE8	Canidae	Canis mesomelas	Black-backed jackal	3	3	52	LE
ZF1	Canidae	Lycaon pictus	African wild dog	2	1	60	EN
ZE9	Canidae	Otocyon megalotis	Bat-eared fox	2	3	52	LE
ZE6	Canidae	Vulpes chama	Cape fox	7	4	46	LE

: (3)

1.2 **Menu Option B**

The user enters the family name of an animal via the keyboard. Display the scientific name, general name, enclosure number and enclosure size for animals belonging to the family name entered by the user, and housed in the ZE area of the zoo.

The **EnclosureNo** field indicates the specific enclosure within an area where the animals are housed, for example in enclosure number ZE5. (ZE represents the area of the zoo and 5 the specific enclosure in that area).

Example of the output of the animals in the *Canidae* family in the ZE area:

ScientificName	GeneralName	EnclosureNo	EnclosureSize
Vulpes chama	Cape fox	ZE6	46
Canis adustus	Side-striped jackal	ZE7	46
Canis mesomelas	Black-backed jackal	ZE8	52
Otocyon megalotis	Bat-eared fox	ZE9	52

(5)

1.3 **Menu Option C**

Display the categories of endangered species and the total number of animal species in each category housed at the zoo. Use a calculated field with the heading **CountAnimals** for the calculation.

Example of the output:

Endangered	CountAnimals
EN	1
LE	37
NE	1
VU	3

(4)

1.4 **Menu Option D**

Display the enclosure number and a calculated field showing the space available for each animal of the different mongoose species indicated in the **GeneralName** field.

Calculate the space available per animal by creating a formula which divides the enclosure size by the total number of animals housed in the enclosure. The calculated values should be displayed with a maximum of two decimal digits. Display the calculated field with the heading **SpacePerAnimal**.

HINT: Use the **NumAdults**, **NumYoung** and **EnclosureSize** fields as part of the formula.

Example of the output of the first five records:

EnclosureNo	SpacePerAnimal
ZD1	6.8
ZD2	5.5
ZD3	3.09
ZD4	10.5
ZD5	3.25
:	

NOTE: The format of the **SpacePerAnimal** field may differ from the given example. (6)

1.5 **Menu Option E**

Increase the number of young animals in the ZF1 enclosure by 3.

If the updating of the record was done successfully, an output message, stating that the record was successfully processed, will be displayed. (4)

1.6 **Menu Option F**

The user is requested to enter the day of the month when the veterinarian (vet) visited the animals, for example 23.

Display the enclosure number, the general name, the date of the visit, the IDs of the particular animals visited and the reason why each of the animals was visited on the specific day of the month entered by the user.

HINT: Use the **DAY()** SQL function in the SQL statement.

Example of the output of the first five records for 23 September 2012:

EnclosureNo	GeneralName	VisitDate	Animal_ID	ReasonForVisit
ZF1	African wild dog	2012-09-23	ZF1_5	Assisted with birth
ZD3	Cape grey mongoose	2012-09-23	ZD3_3	Skin problem
ZD8	Selous' mongoose	2012-09-23	ZD8_1	Routine check-up
ZE6	Cape fox	2012-09-23	ZE6_5	Routine check-up
ZE4	Brown hyena	2012-09-23	ZE4_5	Routine check-up

:

NOTE: The format of the dates in the **VisitDate** field may differ from the given example. (7)

1.7 **Menu Option G**

On 25 September 2012 the veterinarian examined the animal with the ID ZD5_3 in enclosure ZD5 for an ear infection. He indicated that a follow-up visit would be required. Add this data as a new record into the **tblVetVisits** table.

If the record was added to the table successfully, an output message stating that the record was processed successfully will be displayed.

HINT: Use option F to verify the adding of the record to the database. Use 25 September 2012 as input data. (6)

- Enter your examination number as a comment in the first line of the file containing the SQL statements.
- Save your program.
- A printout of the code will be required. **[35]**

QUESTION 2: OBJECT-ORIENTED PROGRAMMING

Animals are transferred to the zoo from time to time. The administrator of the zoo requires software to assist in placing these animals in suitable available enclosures.

The animals at the zoo are classified into three categories according to their size:

- L – Large animal
- M – Medium-sized animal
- S – Small animal

Different animal species are housed in different sized enclosures.

The files required for this question can be found in the folder named **Question2_XXXX** where XXXX refers to the programming language you have studied. You have been provided with a text file named **DataQ2.txt** and an incomplete program that consists of:

- A class unit (Delphi)/object class (Java) which describes the attributes of an enclosure and contains some methods
- A main form unit (Delphi)/test class (Java)

The text file contains the data of an unknown number of enclosures in the zoo.

The details of each enclosure appear on one line with the data items separated by semicolons (;) and hash (#) characters in the following format:

<Type of animal>**;**<Number of animals currently in the enclosure>#<Size of the enclosure in square metres>**;**<Category of animals based on size>#

Example of the data for the first five enclosures in the text file **DataQ2.txt**:

```
Cheetah;3#80.2;L#
Ratel;7#50;S#
XXX;0#20;X#
Serval;5#80.75;M#
XXX;0#36;X#
```

NOTE: XXX denotes an empty enclosure and therefore the category of the animals is unknown and is denoted by an X.

Do the following:

- Rename the given folder for QUESTION 2 by replacing the name of the programming language you have studied with your examination number.

- **Delphi programmers:**

  - Open the given incomplete program file **Question2P.dpr**.
  - Add your examination number as a comment in the first line of both the class unit (**uQuest2**) and the main form unit (**Question2U**).

- **Java programmers:**
  - Open the given incomplete object class **Quest2** and the test class **TestQuestion2**.
  - Add your examination number as a comment in the first line of both the object class (**Quest2**) and the test class (**TestQuestion2**).

- Compile and execute the program. The interface displays three menu options as indicated in the section labelled **QUESTION 2** in **ANNEXURE B (Delphi)/ANNEXURE C (Java)**.

2.1 **Do the following to complete the code in the class unit (Delphi)/object class (Java):**

The given **uQuest2** unit (Delphi)/**Quest2** class (Java) contains the declaration of four attributes for an enclosure object and the set (mutator) and get (accessor) methods for these attributes.

Write code for additional methods as described below.

2.1.1 Write code for a **constructor** method using parameter values to initialise the following attributes:

- Type of animal (fAType/type)
- Total number of this animal type in the enclosure (fNumber/number)
- Size of the enclosure in square metres (fSize/size)
- The category of the animals in the enclosure according to their size, that is L, M or S. (fCat/cat) (4)

2.1.2 Write code for a method named **isSuitable** to determine whether an enclosure is suitable to house a specific group of animals. The method receives as parameters the number of animals the group consists of and their category based on their size. It returns a Boolean value.

The following applies:

- Animals can only be housed in an empty enclosure. An empty enclosure is indicated with "XXX" as animal type.

- The enclosure must be large enough to house the group of animals. The criteria for the size of the enclosures are (on the next page):

- A large animal (L) needs a minimum space of 18 square metres.
- A medium-sized animal (M) needs a minimum space of 12 square metres and a maximum space of less than 18 square metres.
- A small animal (S) needs a minimum space of 7 square metres and a maximum space of less than 12 square metres. (7)

2.1.3 Write code for a **toString** method that will construct and return a string which includes labels and information about the object in the following format:

```
<Animal type>...<Category of the animals in the enclosure>
Enclosure size: <Enclosure size>
Number of animals:  <Number of animals in the enclosure>
<Blank line>
```

Example of the output of the first two objects when their strings are returned by the **toString** method and displayed:

```
Cheetah...L
Enclosure size: 80.2
Number of animals: 3

Ratel...S
Enclosure size: 50.0
Number of animals: 7
```

(4)

2.2 **Do the following to complete the code in the main form unit (Delphi)/test class (Java):**

2.2.1 Declare an array capable of storing 30 enclosure objects and a counter variable to keep track of the number of objects in the array. (2)

2.2.2 Write code to test whether the text file exists.

If the file exists, write code to read lines of text from the text file. For each line of text, extract the data, create an enclosure object and assign the object to the array.

If the text file does not exist, display a suitable message and terminate the program.

NOTE: The objects have to be assigned to the array before the menu options are displayed. (15)

2.2.3 Complete each menu option as follows:

Menu Option A

Write code to display a numbered list of all the information for all the enclosures using the **toString** method.

Example of the output of some of the enclosures:

```
List of all the enclosures
==========================
Enclosure number: 1
Cheetah...L
Enclosure size: 80.2
Number of animals: 3

Enclosure number: 2
Ratel...S
Enclosure size: 50.0
Number of animals: 7

Enclosure number: 3
XXX...X
Enclosure size: 20.0
Number of animals: 0

:
```

(4)

Menu Option B

A number of animals need to be transferred to the zoo and a suitable empty enclosure needs to be identified to house them.

Write code to allow the user to enter the following:

- The type of animal, for example Tiger
- The number of this type of animal
- The category of the animals based on size (for example L, M or S)

Use a conditional loop and the **isSuitable** method to search for a suitable empty enclosure in the array.

If a suitable empty enclosure is found, the attributes of the empty enclosure object in the array must be updated using the relevant set (mutator) methods. A message must be displayed indicating the enclosure number.

HINT: Use menu option A to display all enclosures to see whether the empty enclosure referred to in the message has been updated with the relevant information.

If a suitable empty enclosure is not found, an appropriate message must be displayed.

Test your program with the following test data:

Information for test data set 1:

Animal type: Tiger
Number of animals: 2
Size of the animals: L

Example of the output:

```
These animals were placed in enclosure number 5.

List of all the enclosures
==========================
:

Enclosure number: 5
Tiger...L
Enclosure size: 36.0
Number of animals: 2

:
```

Information for test data set 2:

Animal type: Meerkat
Number of animals: 12
Size of the animals: S

Example of the output:

```
No suitable enclosure was found.
```

(11)

- Make sure that your examination number is entered as a comment in the first line of the class unit (Delphi)/object class (Java) as well as the main form unit (Delphi)/test class (Java).
- Save all the files.
- A printout of the code will be required.
- Print both the class unit (Delphi)/object class (Java) and the main form unit (Delphi)/test class (Java).

[47]

QUESTION 3: PROBLEM–SOLVING PROGRAMMING

The aquarium and the reptile park at the zoo decided to run a competition to inform visitors about their activities. Four displays labelled A, B, C and D have been constructed by the zookeepers in both the aquarium and the reptile park. Visitors take part in the competition by completing an entry form on which they indicate which display they regard as the best.

The information supplied by each participant on his/her entry form is captured as a string using the following format:

`<Section><Display><Gender><Ticket number>`

- **Section** indicates the section housing their favourite display. The section can be either the letter **A** (Aquarium) or the letter **R** (Reptile park).
- **Display** refers to the specific display in the indicated section which they selected as their favourite display. Each display in each section is indicated by the letters A, B, C or D.
- **Gender** refers to the participants' gender. The capital letter **M** or **F** indicates the entry form was filled out by an adult male (M) or adult female (F). The small letter **m** or **f** indicates the entry form was filled out by a boy (m) or a girl (f).
- **Ticket number** refers to the number printed on the entry form.

Examples of the strings that were captured:

- **RCf15:** A girl holding ticket number 15 selected display C of the reptile park (R) section as the best display.
- **ADM33:** An adult male holding ticket number 33 selected display D of the aquarium (A) section as the best display.

The folder **Question3_XXXX** contains a text file **DataQ3_XXXX.txt** where XXXX refers to the programming language you have studied.

The given text file contains the following:

- Code for the declaration of an array named **arrTic**. The array contains 32 strings representing the data captured from the completed entry forms.
- Java code to display a menu with three options.

Content of the **arrTic** array:

`RCm158, ADM33, RCf250, RAf5, BRM32, ADm236, RCm23, RDM54, RCf17, RAm12, ADm9, RCF43, RDm140, RDm23, ACF113, ABf30, RDm22, ARf38, RCF8, RAf53, RCf12, ABF156, ADM31, ADM47, RAf48, ABF246, ABf59, RRM321, ABm36, RCF31, RAm445, ACn26`

Do the following:

- Rename the given folder for QUESTION 3 by replacing the name of the programming language you have studied with your examination number.
- Create a new program/project/application.
- Add your examination number as a comment in the first line of the program file(s) you have created that will contain your code.

  Java programmers: If your solution consists of more than one class file, make sure to add your examination number as a comment in all the class files.
- Save the program file(s) by using the **question number** as part of the filename in the renamed folder for QUESTION 3.
- Develop an interface as follows:
  - **Delphi programmers:**

    Develop an interface to display a menu as indicated in the section labelled **QUESTION 3** in **ANNEXURE B**.
  - **Java programmers:**

    Copy the code from the **DataQ3_Java** text file for displaying a menu as indicated in the section labelled **QUESTION 3** in **ANNEXURE C**.
- Use the following code to complete the "Quit"-menu option:
  - **Delphi programmers:**
    - Application.Terminate;
  - **Java programmers:**
    - System.exit(0);
- Copy the text for the declaration of the array which is supplied in the text file to your program file.
- Complete the code for each menu option as follows:

NOTE: Test your program by running the menu options in sequence, that is option A, then option B and finally option C.

3.1 **Menu Option A**

The data in the given array **arrTic** needs to be validated.

An entry is valid if the following conditions are met for the **first three characters**:

- The first character indicating the section of choice can only be the letter **A** or the letter **R**.
- The second character indicating the display of choice can only be the letters **A**, **B**, **C** or **D**.
- The third character indicating the gender can only be the letters **M**, **F**, **m** or **f**.

Write code to identify invalid entries in the **arrTic** array. For each invalid entry:

- Display the invalid entry
- Replace the invalid entry in the **arrTic** array with the letter '**Z**'.

Example of the output:

```
Invalid entries:
BRM32
ARf38
RRM321
ACn26
```

(8)

3.2 **Menu Option B**

The program has to randomly select a number. The number represents the position of the winning ticket from the array.

Display the randomly generated number and the code of the winning ticket with suitable labels.

If an invalid entry is selected the program has to display the word 'Invalid' and continue selecting tickets randomly until a valid entry is selected.

NOTE:

- You may assume that menu option A has been selected and executed already.
- Due to the random function, the number generated by your program may be different from those shown in the examples below.

Example 1:

```
The position of the winning ticket in the array: 24
The winning ticket: ABM47
```

Example 2:

If a few invalid entries were randomly selected before a valid winning number was selected, the output would be as follows:

```
Invalid
Invalid
The position of the winning ticket in the array: 6
The winning ticket: ADm236
```

(11)

3.3 **Menu Option C**

Management wants to award medals to the three displays indicated as the best by the participants. They need a report showing the three medal-winning displays.

Points are awarded to the displays as follows:

- A display indicated as the best by an adult is awarded 12 points.
- A display indicated as the best by a child is awarded 5 points.

The medals are awarded as follows:

- The display with the highest total points is awarded the gold medal.
- The display with the second highest total points is awarded the silver medal.
- The display with the third highest total points is awarded the bronze medal.

Only data from valid ticket entries must be used to compile the report.

NOTE: An invalid entry is indicated by the single character 'Z' in the array after option A has been executed.

Example of the output:

```
Medal winning displays:
Medal         Display    Points
Gold          RC         61
Silver        AD         46
Bronze        AB         39
```

(19)

- Make sure your examination number is entered as a comment in the first line of any program files containing your code.
- Save all the files.
- A printout for the code will be required.

[38]

TOTAL: 120

ANNEXURE A: DATABASE STRUCTURE AND SAMPLE DATA

This annexure shows the data structure and sample data for the tables used in the **ZooDB** database in **QUESTION 1.**

tblCarnivores: This table contains data on all the carnivores housed at the zoo.

Table structure:

Field Name	Type	Size	Description
EnclosureNo	Text	4	A unique code assigned to each enclosure
FamilyName	Text	15	The scientific classification of the animal species housed in the enclosure
ScientificName	Text	30	The scientific name for the animals housed in the enclosure
GeneralName	Text	30	The general name for the animals housed in the enclosure
NumAdults	Number	Integer	The number of full-grown animals housed in the enclosure
NumYoung	Number	Integer	The number of young animals (not fully grown) housed in the enclosure
EnclosureSize	Number	Integer	The size of the enclosure in square metres
Endangered	Text	2	The endangered species category used. The categories are: LE – Least endangered means there are currently no identifiable risks to the species. VU – Vulnerable means the species is facing a high risk of extinction in the wild. NE – Not endangered means it does not meet any of the criteria that would categorise it as risking extinction but it is likely to do so in future. EN – Endangered means the species is facing an extremely high risk of extinction in the wild.

Sample data:

EnclosureNo	FamilyName	ScientificName	GeneralName	NumAdults	NumYoung	EnclosureSize	Endangered
ZA1	Felidae	Acinonyx jubatus	Cheetah	2	1	50	VU
ZA2	Felidae	Caracal caracal	Caracal	2	2	36	LE
ZA5	Felidae	Felis nigripes	Black-footed cat	2	0	36	VU
ZA9	Felidae	Felis silvestris	Wildcat	3	3	36	LE
ZB2	Felidae	Leptailurus serval	Serval	3	5	80	LE
ZB5	Felidae	Panthera pardus	Leopard	3	3	700	LE
ZB6	Felidae	Panthera leo	Lion	4	2	800	VU
ZB9	Viverridae	Civettictis civetta	African civet	3	0	50	LE
ZC2	Viverridae	Genetta tigrina	Cape genet	8	1	50	LE

tblVetVisits: This table contains data on the veterinarian's visits to some of the animals during one week of the month of September.

Table structure:

Field Name	Type	Size	Description
VisitID	Autonumber	Long integer	Number to uniquely identify every visit by the veterinarian
VisitDate	Date/Time	Short date	Date on which the veterinarian visited enclosure (yyyy/mm/dd)
EnclosureNo	Text	4	Code for enclosure that was visited
ReasonForVisit	Text	30	Reason why the veterinarian visited the animal in the enclosure
FollowUp	Yes/No		Was a follow-up visit scheduled? (Yes/No)
Animal_ID	Text	10	Code to uniquely identify the animal in the enclosure that was visited

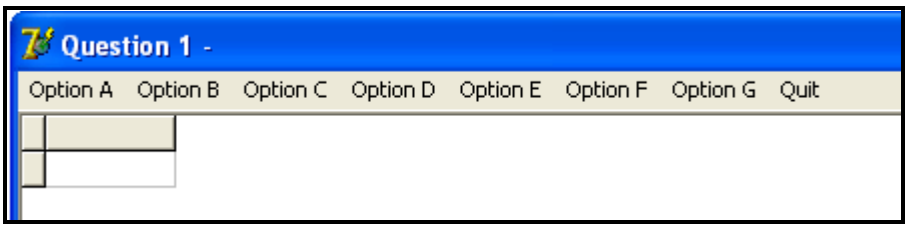
Sample data:

VisitID	VisitDate	EnclosureNo	ReasonForVisit	FollowUp	Animal_ID
1	2012/09/20	ZA1	Skin problem	☑	ZA1_9
2	2012/09/20	ZC2	Routine check-up	☐	ZC2_3
3	2012/09/20	ZC3	Injured eye	☑	ZC3_8
4	2012/09/20	ZD4	Routine check-up	☐	ZD4_2
5	2012/09/20	ZA5	Routine check-up	☐	ZA5_2
6	2012/09/20	ZB6	Routine check-up	☐	ZB6_6
7	2012/09/20	ZA9	Ear infection	☑	ZA9_2

ANNEXURE B: DELPHI – GUI INTERFACES PER QUESTION

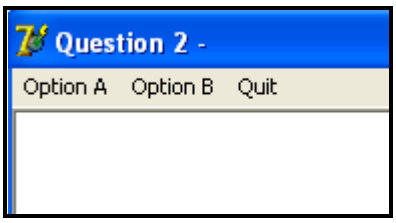
QUESTION 1

When you execute the program, the interface below will be displayed.



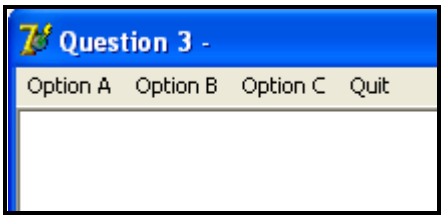
QUESTION 2

When you execute the program, the interface below will be displayed.



QUESTION 3

You are required to create the following interface as part of the solution for QUESTION 3. When you execute the program, the interface below must be displayed.

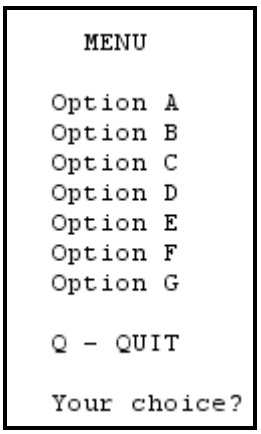


NOTE: Use the **MainMenu** component to create the menu.

ANNEXURE C: JAVA – GUI INTERFACES PER QUESTION

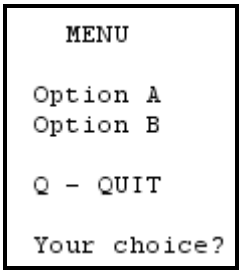
QUESTION 1

When you execute the program, the interface below will be displayed.



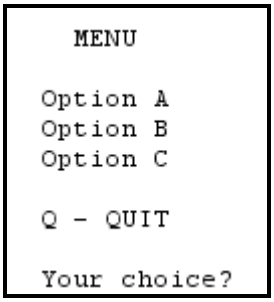
QUESTION 2

When you execute the program, the interface below will be displayed.



QUESTION 3

Copy and use the code from the **DataQ3_Java** text file to create the following interface as part of the solution for QUESTION 3. When you execute the program, the interface below must be displayed.



ANNEXURE D: DELPHI – TROUBLESHOOTING DATABASE PROBLEMS

D.1 If you cannot use the given database:

- Create your own database with the name **ZooDB** that includes a table named **tblCarnivores** and another table named **tblVetVisits** in the same folder as your program for QUESTION 1.
- Import the two text files (**tblCarnivores.txt** and **tblVetVisits.txt**) to use as data for the different tables.
- The first line in the text files contains the field names to be used.

D.2 If your program cannot establish connectivity with the database:

- Make sure that the database file **ZooDB** is in the same folder as your program for QUESTION 1. If this is not the case, copy the database file into the same folder as your program.

D.3 If your program establishes a connectivity but no data is displayed:

- Click on the ADOQuery component named **qryRec**.
- Click on the Ellipsis button (three dots) to the right of the 'ConnectionString' property in the Object Inspector.
- Click on the Build button which takes you to the Data Link Properties dialogue box.
- Click on the Provider tab to open the Provider tab sheet and select Microsoft Jet 4.0 OLE DB Provider. Click on the Next button.
- The Connection tab sheet will be displayed. The first option on the Connection tab sheet provides an Ellipsis button (three dots) that allows you to browse and look for the **ZooDB** file. You will find this file in the folder for QUESTION 1. Once you have found it, select the **ZooDB** file and then click on the Open button.
- Remove the user name Admin.
- Click on the Test Connection button.
- Click OK on each of the open dialogue windows.

ANNEXURE E: JAVA – TROUBLESHOOTING DATABASE PROBLEMS

E.1 If you cannot use the given database:

- Create your own database with the name **ZooDB** that includes a table named **tblCarnivores** and another table named **tblVetVisits** in the same folder as your program for QUESTION 1.
- Import the two text files (**tblCarnivores.txt** and **tblVetVisits.txt**) to use as data for the different tables.
- The first line in the text files contains the field names to be used.

E.2 If your program cannot establish connectivity with the database:

- Make sure that the database file **ZooDB** is in the same folder as your program for QUESTION 1. If this is not the case, copy the database file into the same folder as your program.

E.3 If you cannot establish connectivity with the given database with the given program files, use the following source code to ensure database connectivity:

```
try
  {
     Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
     String filename = "ZooDB.mdb";
     String database = "jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ=";
            database += filename.trim () + ";DriverID=22;READONLY=true}";
     Connection conn = DriverManager.getConnection (database, "", "");
  }
  catch (Exception e)
  {
     System.out.println ("Unable to connect to the database");
  }
```