



basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

**NATIONAL
SENIOR CERTIFICATE**

GRADE 12

INFORMATION TECHNOLOGY P1

NOVEMBER 2012

MEMORANDUM

MARKS: 120

The memorandum consists of 32 pages.

GENERAL INFORMATION

- These marking guidelines are to be used as the basis for the marking session. They were prepared for use by markers, all of whom are required to attend a rigorous standardisation meeting to ensure that the guidelines are consistently interpreted and applied in the marking of candidates' scripts.
- It is acknowledged that there may be different views about some matters of emphasis or detail in the guidelines, and different interpretations of the application thereof.
- Note that learners who provide an alternate correct solution to that given in the marking guidelines will be given full credit for the relevant question.
- **ANNEXURES A, B and C** (pages 3–10) include the marking grid for each question for using either one of the two programming languages.
- **ANNEXURES D, E and F** (pages 11–19) contain the solutions for Delphi for QUESTIONS 1 to 3 in programming code.
- **ANNEXURES G, H, I and J** (pages 16–28) contain the solutions for Java for QUESTIONS 1 to 3 in programming code.
- Copies of ANNEXURES A, B and C (pages 3–6) should be made for each learner and completed during the marking session.

ANNEXURE A:**QUESTION 1: MARKING GRID – PROGRAMMING AND DATABASE****GENERAL NOTES:**

- Only penalise for the incorrect use of quotes once. Repeated incorrect use of quotes in follow up questions doesn't get penalised.
- The use of = for strings, the use of LIKE may be used as alternative

CENTRE NUMBER:		EXAMINATION NUMBER:	
QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
1.1	Query: Correct fields (or *) ✓; correct table ✓; correct ORDER BY both fields ✓	3	
	SQL: SELECT * FROM tblCarnivores ORDER BY FamilyName, ScientificName		
1.2	Query: Correct fields & table ✓; Correct WHERE clause displaying the correct family using input variable ✓ AND ✓ EnclosureNo starting with ZE ✓ using LIKE ✓	5	
	SQL(D): SELECT ScientificName, GeneralName, EnclosureNo, EnclosureSize FROM tblCarnivores WHERE EnclosureNo LIKE "ZE%" AND FamilyName = " + sX + "'		
	Alternative: ...FamilyName LIKE "%' + sX + '%" In Delphi accept Parameters wit SQL.		
1.3	Query: Correct field & table ✓; COUNT(*) ✓ AS CountAnimals ✓; GROUP BY Endangered ✓	4	
	SQL: SELECT Endangered, Count(*) AS CountAnimals FROM tblCarnivores GROUP BY Endangered		
	Alternative: Count(Endangered) Don't penalise for using Distinct		

<p>1.4</p>	<p>Query: Correct field & table ✓; <i>SpacePerAnimal</i> ✓ correctly calculated with brackets ✓; ROUND or FORMAT to 1 or 2 dec ✓; correct WHERE clause testing <i>GeneralName</i> for <i>mongoose</i> ✓ with LIKE ✓</p> <hr/> <p>SQL(D): SELECT EnclosureNo, Format(EnclosureSize / (NumAdults+NumYoung), '#.0#') AS SpacePerAnimal FROM tblCarnivores WHERE GeneralName LIKE "%mongoose"</p> <p>Alternative: Format(EnclosureSize/(NumAdults+NumYoung), '#.00') Format(EnclosureSize/(NumAdults+NumYoung), "0.00") Format(EnclosureSize/(NumAdults+NumYoung), ".00") Round(EnclosureSize/(NumAdults+NumYoung), 2)</p> <p>Also accept the use of ScientificName="Herpestidae"</p> <hr/> <p>SQL(J): SELECT EnclosureNo, Format(EnclosureSize / (NumAdults+NumYoung), '#.0#') AS SpacePerAnimal FROM tblCarnivores WHERE GeneralName LIKE '%mongoose'</p> <p>Alternative: Format(EnclosureSize/(NumAdults+NumYoung), '#.00') Format(EnclosureSize/(NumAdults+NumYoung), '0.00') Format(EnclosureSize/(NumAdults+NumYoung), '.00') Round(EnclosureSize/(NumAdults+NumYoung), 2)</p> <p>Also accept the use of ScientificName='Herpestidae'</p>	<p>6</p>	
<p>1.5</p>	<p>Query: UPDATE correct table ✓; SET the correct field ✓ with a formula increasing the value with 3 ✓; WHERE correct EnclosureNo ✓</p> <p>NOTE: the use of the same numerical field on both sides of the = sign for the formula.</p> <hr/> <p>SQL(D): UPDATE tblCarnivores SET NumYoung = NumYoung + 3 WHERE EnclosureNo = "ZF1"</p> <hr/> <p>SQL(J): UPDATE tblCarnivores SET NumYoung = NumYoung + 3 WHERE EnclosureNo = 'ZF1'</p>	<p>4</p>	

QUESTION 1: MARKING GRID – PROGRAMMING AND DATABASE (continued)

<p>1.6</p>	<p>Query: SELECT correct fields✓; FROM both tables✓; WHERE clause linking both tables on EnclosureNo ✓(left side =) ✓(right side =); using DAY✓ function on visitDate✓; with variable ✓</p> <p>SQL(D): SELECT tblVetVisits.EnclosureNo, GeneralName, VisitDate,ReasonForVisit, Animal_ID FROM tblCarnivores, tblVetVisits WHERE Day(VisitDate)='+sX+' AND tblCarnivores.EnclosureNo = tblVetVisits.EnclosureNo</p> <p>Alternative: Use aliases for tables names: SELECT C.EnclosureNo, GeneralName, VisitDate,ReasonForVisit, Animal_ID FROM tblCarnivores C, tblVetVisits V WHERE Day(VisitDate)='+sX+' AND C.EnclosureNo = V.EnclosureNo</p> <p>Alternative: Using JOIN notation: SELECT tblCarnivores.EnclosureNo, tblCarnivores.GeneralName, tblVetVisits.VisitDate, tblVetVisits.ReasonForVisit, Animal_ID FROM tblCarnivores INNER JOIN tblVetVisits ON tblCarnivores.EnclosureNo = tblVetVisits.EnclosureN WHERE Day(visitDate)='+Sx</p> <p>NOTE: INNER JOIN may be replaced by LEFT or RIGHT JOIN</p>		
	<p>SQL(J): SELECT tblVetVisits.EnclosureNo, GeneralName, VisitDate,ReasonForVisit, Animal_ID FROM tblCarnivores, tblVetVisits WHERE Day(VisitDate)="+sX+" AND tblCarnivores.EnclosureNo = tblVetVisits.EnclosureNo</p> <p>Alternative: Use aliases for tables names: SELECT C.EnclosureNo, GeneralName, VisitDate,ReasonForVisit, Animal_ID FROM tblCarnivores C, tblVetVisits V WHERE Day(VisitDate)="+sX+" AND C.EnclosureNo = V.EnclosureNo</p> <p>Alternative: Use aliases for tables names AND CORRECT DATATYPE SELECT C.EnclosureNo, GeneralName, VisitDate,ReasonForVisit, Animal_ID FROM tblCarnivores C, tblVetVisits V WHERE Day(VisitDate)=""+sX+" AND C.EnclosureNo = V.EnclosureNo</p> <p>Alternative: Using JOIN notation: SELECT tblCarnivores.EnclosureNo, tblCarnivores.GeneralName, tblVetVisits.VisitDate, tblVetVisits.ReasonForVisit, Animal_ID FROM tblCarnivores INNER JOIN tblVetVisits ON tblCarnivores.EnclosureNo = tblVetVisits.EnclosureN WHERE Day([visitDate])="+sX</p> <p>NOTE: INNER JOIN may be replaced by LEFT or RIGHT JOIN</p>	<p>7</p>	

QUESTION 1: MARKING GRID – PROGRAMMING AND DATABASE (continued)

1.7	<p>Query: INSERT INTO correct table✓; list 5 fields (not [VisitID] autonumber field)✓; Values in correct order as listed in fields✓; date value using #2012/10/25#✓; all text fields values✓; boolean field value✓</p> <p>NOTE: If no fields listed but six values listed (1 mark ½)</p>	6	
	<p>SQL(D): INSERT INTO tblVetVisits (VisitDate, EnclosureNo, ReasonForVisit, FollowUp, Animal_ID) VALUES (#2012/10/25#, "ZD5", "Ear infection", True, "ZD5_3")</p> <p>Accept: yes/on/1 instead of true The use of " " for the date in the correct format (short date)</p>		
	<p>SQL(J): INSERT INTO tblVetVisits (VisitDate, EnclosureNo, ReasonForVisit, FollowUp, Animal_ID) VALUES (#2012/10/25#, 'ZD5', 'Ear infection', true, 'ZD5_3')</p> <p>Accept: yes/on/1 instead of true The use of ' ' for the date in the correct format (short date)</p>		
	TOTAL:		
		35	

ANNEXURE B:**QUESTION 2: MARKING GRID – OBJECT-ORIENTED PROGRAMMING****GENERAL NOTES:**

- If the learner changed any given data type (e.g. character to string) penalise with ONE mark.
- Syntax error (e.g. ;) penalise only ONCE.
- In Java the use of single = in stead of == penalise only ONCE.

CENTRE NUMBER:		EXAMINATION NUMBER:	
QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
2.1.1	<p>Parameterised constructor FOUR correct parameters✓ with correct corresponding data type✓ Assign/set four parameters ✓✓</p> <p>NOTE: if all four assignment statements in incorrect order (max 1 mark)</p>	4	
2.1.2	<p>isSuitable METHOD: Parameters: number of animals and size category✓ Return a boolean ✓ Test if empty enclosure ✓; Calculate size✓ Test if suitable for animal ✓ (large/medium/small) Use nested IF/case/switch method with intervals using 7, 12 and 18 in logical format✓ (If using separate IF's a return variable must be used) Assign a return value/answer✓</p>	7	
2.1.3	<p>toString METHOD: Display type and category✓; Display two labels ✓; Display two numerical values: real/double enclosure size and number of animals✓; Display over three lines (Delphi: ...#13;...#10) (Java: ...\\n...\\n...\\n) ✓</p> <p>NOTE: Are allowed to call methods</p> <p>NOTE – Penalise with ONE mark if:</p> <ul style="list-style-type: none"> • No value is returned • Any incorrect variables used 	4	
2.2.1	<p>Declares object array – size 30 or more ✓; Counter for array✓</p>	2	

2.2.2	<p>INITIALISATION OF ARRAY: Test if file exist ✓ may use a IF...then...else/try... catch(FileNotFoundException e) (all statements must be included in the correct place)</p> <p><i>File doesn't exists:</i> Appropriate message✓; Terminate program/event/exit✓</p> <p><i>File does exists:</i> {DELPHI: AssignFile, Reset JAVA: Create object to read from file} ✓✓; Initialise array counter ✓(also accept at declaration); Loop: while not eof/hasNext()✓; Read line from text file✓; Extract the four field values/variable values correctly (1 mark per field/variable using e.g. copy/pos/split/indexOf) ✓✓✓✓; Increment array counter ✓; Instantiate object using parameterized constructor ✓✓ (one mark left side = and one mark right side. Check correct parameter order from constructor)</p>	15	
2.2.3	<p>MENU OPTION A: Heading✓; Loop through array✓; Display enclosure number list✓ Using toString method display enclosure information✓</p>	4	
	<p>MENU OPTION B: Input type, number, category of animal✓; Initialize variables (counter)✓; Conditional loop✓ (inside array range AND flag)✓ IF statement calling isSuitable method✓ with correct order and matching parameters ✓ If found: Set 3 attributes at correct counter position in array using set methods ✓; Change flag ✓;</p> <p>Increment counter outside the if (inside the while)✓</p> <p><i>Outside loop:</i> Display message with enclosure number✓ (also accept if inside the IF...Then, inside the While loop) Display message if suitable enclosure was not found ✓</p> <p>NOTE: The use of a flag/break/exit in the correct place is acceptable. Incorrect placement of messages (maximum 1 mark)</p>	11	
	TOTAL:	47	

ANNEXURE C:**QUESTION 3: MARKING GRID – PROBLEM-SOLVING PROGRAMMING****GENERAL NOTES:**

- If the learner changed any given data type (e.g. character to string) penalise with ONE mark.
- Syntax error (e.g. ;) penalise only ONCE.
- In Java the use of single = in stead of == penalise only ONCE.
- In Java accept the use of the Scanner class instead of BufferedReader

CENTRE NUMBER:		EXAMINATION NUMBER:	
QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
3.1	<p>MENU OPTION A: Loop through array ✓; Validate first character: A or R ✓ Validate second character: A, B, C or D ✓ Validate third character: M, F, m or f ✓;</p> <p>Correct use of logical operators (AND/OR) in validating ✓✓ (or using nested if...then...else or separate IF statements or replace the invalid entries with Z and exit/break or using flag/variable)</p> <p>Display invalid entries ✓; Replace invalid entries with Z in array at correct index✓</p>	8	
3.2	<p>MENU OPTION B: Generate random number✓; Number in range of number of array elements (1-32 in Delphi or 0-31 in Java) ✓; Loop✓; Correct use of loop conditions✓; Check if valid ticket ✓; <i>IF valid:</i> stop the loop✓; <i>IF not valid:</i> generate another random value✓; Display message for invalid entry✓</p> <p><i>Outside loop:</i> Display message✓ and winning number✓; (also accept if inside the IF...Then, inside the While loop) Display winning ticket string from array✓</p>	11	

3.3	<p>MENU OPTION C: Declare data structure(s)✓ for category elements✓ of any data types (e.g. arrays for storing displays (AA, AB ...) and points awarded to each display; not 32 elements) <i>Calculating the points:</i> Nested loops: loop through points array (8 elements)✓ and loop through arrTic✓ (32 elements) in any order; IF valid ticket✓; check first two characters✓ of ticket vs category (AA, BB...)✓; check for adult/child✓; Increment points correctly for correct category (AA, BB...) ✓ <i>Sort both arrays:</i> Nested Loops✓ using correct indices for data structure declared✓; IF with correct condition ✓; Swop array values ✓; simultaneously swop the second value ✓ correct data type for temporary variable ✓ <i>Display results:</i> Heading(s)✓; Display output in columns/tabs✓; Display medals label (gold/silver/bronze)✓, correct category, and corresponding points value for that category✓</p>	19	
TOTAL:		38	

SUMMARY OF LEARNER'S MARKS:

	QUESTION 1	QUESTION 2	QUESTION 3	GRAND TOTAL
MAX. MARKS	35	47	38	120
LEARNER'S MARKS				

ANNEXURE D: SOLUTION FOR QUESTION 1: DELPHI

```

unit Question1_UMEMO;
//Solution for Question 1
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, DB, ADODB, Grids, DBGrids, ExtCtrls, Buttons, Menus;
type
  TfrmRec = class(TForm)
    qryRec: TADOQuery;
    dsrQry: TDataSource;
    grdRec: TDBGrid;
    mnuMain: TMainMenu;
    mnuOptionA: TMenuItem;
    mnuOptionB: TMenuItem;
    mnuOptionC: TMenuItem;
    mnuOptionD: TMenuItem;
    mnuOptionE: TMenuItem;
    mnuOptionF: TMenuItem;
    mnuOptionG: TMenuItem;
    mnuQuit: TMenuItem;
    procedure mnuOptionAClick(Sender: TObject);
    procedure mnuOptionBClick(Sender: TObject);
    procedure mnuOptionCClick(Sender: TObject);
    procedure mnuOptionDClick(Sender: TObject);
    procedure mnuOptionEClick(Sender: TObject);
    procedure mnuOptionFClick(Sender: TObject);
    procedure mnuOptionGClick(Sender: TObject);
    procedure mnuQuitClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  frmRec: TfrmRec;
implementation
{$R *.dfm}
//=====
procedure TfrmRec.mnuOptionAClick(Sender: TObject);
begin
  qryRec.Close;
  qryRec.SQL.Text:='SELECT * FROM tblCarnivores ORDER BY FamilyName,
ScientificName';
  qryRec.Open;
end;
//=====
procedure TfrmRec.mnuOptionBClick(Sender: TObject);
var
  sX : String;
begin
  sX := INPUTBOX('Question 1', 'Enter the family name', 'Canidae');
  qryRec.Close;
  qryRec.SQL.Text:= 'SELECT ScientificName, GeneralName, EnclosureNo,
EnclosureSize '+
    'FROM tblCarnivores '+
    'WHERE (FamilyName LIKE "%'+sX+'%" AND (EnclosureNo LIKE "ZE%"))';
  qryRec.Open;
end;
//=====
procedure TfrmRec.mnuOptionCClick(Sender: TObject);
begin

```

```

    qryRec.Close;
    qryRec.SQL.Text := 'SELECT Endangered, Count(NumAdults+NumYoung) AS
CountAnimals '+
                        'FROM tblCarnivores ' +
                        'GROUP BY Endangered';

    qryRec.Open;
end;
//=====
procedure TfrmRec.mnuOptionDClick(Sender: TObject);
begin
    qryRec.Close;
    qryRec.SQL.Text := 'SELECT EnclosureNo, Format((EnclosureSize/
(NumAdults+NumYoung)), "#.0#") AS SpacePerAnimal '+
                        'FROM tblCarnivores '+
                        'WHERE GeneralName LIKE "%mongoose%" ';

    qryRec.Open;
end;
//=====
procedure TfrmRec.mnuOptionEClick(Sender: TObject);
begin
    qryRec.Close;
    qryRec.SQL.Text := 'UPDATE tblCarnivores ' +
                        'SET NumYoung = NumYoung + 3 ' +
                        'WHERE EnclosureNo="ZF1"';

    qryRec.ExecSQL;
    MessageDlg('Record Processed Successfully', mtInformation, [mbOk], 0);
end;
//=====
procedure TfrmRec.mnuOptionFClick(Sender: TObject);
var
    sX : String;
begin
    sX := INPUTBOX('Question 1', 'Enter the day of the month when the visit took
place e.g. 23', '23');
    qryRec.Close;
    qryRec.SQL.Text := 'SELECT C.EnclosureNo, GeneralName, VisitDate,
ReasonForVisit, Animal_ID ' +
                        'FROM tblCarnivores C, tblVetVisits V ' +
                        'WHERE Day(VisitDate)=' + sX + ' AND C.EnclosureNo = V.EnclosureNo';

    qryRec.Open;
end;
//=====
procedure TfrmRec.mnuOptionGClick(Sender: TObject);
begin
    qryRec.Close;
    qryRec.SQL.Text := 'INSERT INTO tblVetVisits ' +
                        '(VisitDate, EnclosureNo, ReasonForVisit, FollowUp,
Animal_ID) '+
                        'VALUES (#2012/10/25#, "ZD5", "Ear infection", True,
"ZD5_3")';

    qryRec.ExecSQL;
    MessageDlg('Record Processed Successfully', mtInformation, [mbOk], 0);
end;
//=====
procedure TfrmRec.mnuQuitClick(Sender: TObject);
begin
    Application.Terminate;
end;

end.

```

ANNEXURE E: SOLUTION FOR QUESTION 2: DELPHI**QUEST2 CLASS UNIT**

```
unit uQuest2_Memo;
{*** Solution for class unit of question 2 ***}
interface

TYPE
  TQuest2 = class(TObject)
  private
    fAType      : String;
    fNumber     : Integer;
    fSize       : Real;
    fCat        : Char;
  public
    constructor create(sAType: String;iNum: integer;rSize: Real;cCat: Char);
    function toString:String;
    function isSuitable(cCat:char; iNumber:integer):Boolean;
    procedure setAType(sAType : String);
    procedure setNumber(iNumber : Integer);
    procedure setSize(rSize : Real);
    procedure setCat(cCat : Char);
    function getAType:String;
    function getNumber:integer;
    function getSize:real;
    function getCat:Char;
  end;

implementation

uses SysUtils;

{ TQuest2 }

constructor TQuest2.create(sAType: String;iNum: integer;rSize: Real;cCat:
Char);
begin
  fAType := sAType;
  fNumber := iNum;
  fSize := rSize;
  fCat := cCat;
end;

function TQuest2.isSuitable(cCat:char; iNumber:integer):Boolean;
var
  rSpace :real;
begin
  Result := false;
  if fAType = 'XXX' then
  begin
    rSpace := fSize / iNumber;
    case cCat of
      'L': Result := rSpace >= 18;
      'M': Result := (rSpace >= 12) and (rSpace < 18);
      'S' : Result := (rSpace >= 7) and (rSpace < 12);
    end;
  end;
end;

function TQuest2.toString:String;
begin
```

```
    Result := fAType + '...' + fCat + #13 + 'Enclosure size: ' +
        FloatToStrF(fSize, ffFixed, 8,1) + #13 + 'Number of animals: ' +
        IntToStr(fNumber) + #13 + #13;
end;

procedure TQuest2.setAType(sAType: String);
begin
    fAType := sAType;
end;

procedure TQuest2.setSize(rSize: Real);
begin
    fSize := rSize;
end;

procedure TQuest2.setCat(cCat: Char);
begin
    fCat := cCat;
end;

procedure TQuest2.setNumber(iNumber: Integer);
begin
    fNumber := iNumber;
end;

function TQuest2.getAType:String;
begin
    Result := fAType;
end;

function TQuest2.getNumber:integer;
begin
    Result := fNumber;
end;

function TQuest2.getSize:real;
begin
    Result := fSize;
end;

function TQuest2.getCat:Char;
begin
    Result := fCat;
end;

end.
```

MAIN FORM UNIT

```
unit Question2U_Memo;
{*** Solution for main unit of question 2 ***}

interface

uses
    Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
    Dialogs, StdCtrls, ComCtrls, Menus,
    uQuest2_Memo;

type
    TfrmQ2 = class(TForm)
        mnuMain: TMainMenu;
        mnuOptionA: TMenuItem;
```

```
mnuQuit: TMenuItem;
redQ2: TRichEdit;
mnuOptionB: TMenuItem;
procedure mnuQuitClick(Sender: TObject);
procedure mnuOptionbClick(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure mnuOptionAClick(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;

var
  frmQ2: TfrmQ2;

implementation

var
  EnclosuresArr :array[1..30] of TQuest2;
  iCount :integer;

{$R *.dfm}
{$R+}

procedure TfrmQ2.FormCreate(Sender: TObject);
var
  TFile : TextFile;
  iPos, iNumber : integer;
  rSize :real;
  cCat :Char;
  sLine, sAnimal :String;
begin
  if FileExists ('DataQ2.txt') <> true then
    begin
      ShowMessage('File does not exist');
      Exit;
    end;
  AssignFile(TFile, 'DataQ2.txt');
  Reset(TFile);

  iCount := 0;
  while NOT EOF(TFile) AND (iCount < 30) do
    begin
      inc(iCount);
      readln(TFile, sLine);
      iPos := pos(';', sLine);
      sAnimal := copy(sLine, 1, iPos -1);
      delete(sLine, 1, iPos);

      iPos := pos('#', sLine);
      iNumber := StrToInt(copy(sLine, 1, iPos -1));
      delete(sLine, 1, iPos);

      iPos := pos(';', sLine);
      rSize := StrToFloat(copy(sLine, 1, iPos -1));
      delete(sLine, 1, iPos);

      cCat := sLine[1];

      EnclosuresArr[iCount] := TQuest2.create(sAnimal, iNumber, rSize, cCat);
    end;
  closeFile(TFile);
```

```
end;

procedure TfrmQ2.mnuOptionAClick(Sender: TObject);
var
  K :integer;
begin
  redQ2.Lines.Add('List of all enclosures');
  redQ2.Lines.Add('=====');
  For K := 1 to iCount do
    begin
      redQ2.Lines.Add('Enclosure number: ' + IntToStr(K) + #13 +
EnclosuresArr[K].toString);
    end;
  end;

procedure TfrmQ2.mnuOptionBClick(Sender: TObject);
var
  K,iNum :integer;
  bFound :boolean;
  cCat   :char;
  sAType :String;
begin
  sAType := InputBox('Animal type', 'Enter the type of animal for example
Tiger','Tiger');
  iNum := StrToInt(InputBox('Number of animals', 'Enter the number of
animals','2'));
  cCat := InputBox('Category', 'Enter the category (L/M/S)', 'L')[1];
  bFound := false;
  K := 1;
  While (bFound <> true) and (K <= iCount) do
    begin
      if EnclosuresArr[K].isSuitable(cCat, iNum) then
        begin
          EnclosuresArr[K].setAType(sAType);
          EnclosuresArr[K].setCat(cCat);
          EnclosuresArr[K].setNumber(iNum);
          bFound := true;
        end
      else
        inc(K);
      end;
    redQ2.Lines.Clear;
    if NOT(bFound) then
      redQ2.Lines.Add('No suitable enclosure was found')
    else
      begin
        redQ2.Lines.Clear;
        redQ2.Lines.Add('These animals were placed in enclosure number ' +
IntToStr(K));
        redQ2.Lines.Add(' ');
        mnuOptionA.Click;
      end;
  end;

procedure TfrmQ2.mnuQuitClick(Sender: TObject);
begin
  Application.Terminate;
end;
end.
```


ANNEXURE F: SOLUTION FOR QUESTION 3: DELPHI

```

unit Question3U_MEMO;
//Solution for Question 3...
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Buttons, StdCtrls, ComCtrls, Menus;

type
  TfrmQuestion3 = class(TForm)
    MainMenu: TMainMenu;
    mnuOptionA: TMenuItem;
    mnuOptionB: TMenuItem;
    mnuOptionC: TMenuItem;
    mnuQuit: TMenuItem;
    redQ3: TRichEdit;
    procedure mnuOptionAClick(Sender: TObject);
    procedure mnuQuitClick(Sender: TObject);
    procedure mnuOptionBClick(Sender: TObject);
    procedure mnuOptionCClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmQuestion3: TfrmQuestion3;
  arrTic : Array[1..32] of string =
    ('RCm158', 'ADM33', 'RCf250', 'RAf5', 'RRM32', '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');

implementation

{$R *.dfm}
{$R+}

var
  //arrays used in solution for Option C
  arrDisplay : Array[1..8] of string =
    ('AA', 'AB', 'AC', 'AD', 'RA', 'RB', 'RC', 'RD');
  arrPoints : Array[1..8] of Integer = (0,0,0,0,0,0,0,0);
  arrMedal : Array[1..3] of string = ('Gold', 'Silver', 'Bronze');

procedure TfrmQuestion3.FormCreate(Sender: TObject);
begin
  Randomize;
end;

procedure TfrmQuestion3.mnuOptionAClick(Sender: TObject);
var
  A : Integer;
begin
  redQ3.Lines.Clear;
  redQ3.Lines.Add('Invalid entries:');
  For A := 1 to 32 do
    IF (arrTic[A][1] in ['A','R']) AND
        (arrTic[A][2] in ['A'..'D']) AND

```

```
(upCase(arrTic[A][3]) in ['M','F'])
then      //valid ticket
else
  begin //invalid ticket
    redQ3.Lines.Add(arrTic[A]);
    arrTic[A] := 'Z';
  end;
end;

procedure TfrmQuestion3.mnuOptionBClick(Sender: TObject);
var
  iTicket :integer;
  bValid  :boolean;
begin
  bValid := false;
  iTicket := random(32) + 1;
  while (bValid = false) do
  begin
    if arrTic[iTicket] = 'Z' then
    begin
      bValid := false ;
      redQ3.Lines.Add('Invalid');
      iTicket := random(32) + 1;
    end
    else
      bValid := true;
    end;
  redQ3.Lines.Add('The position of the winning ticket in the array: ' +
    intToStr(iTicket));
  redQ3.Lines.Add('The winning ticket: ' + arrTic[iTicket] );
end;

procedure TfrmQuestion3.mnuOptionCClick(Sender: TObject);
var
  A, D, iPoint, iTemp : Integer;
  sTemp                : string;
begin
  //For each display calculate the number of points:
  for D := 1 to 8 do
    arrPoints[D] := 0;

  for D := 1 to 8 do
  begin
    For A := 1 to 32 do
    begin
      IF pos(arrDisplay[D], arrTic[A]) = 1 //only valid tickets <> Z
      then
        begin
          case arrTic[A][3] of
            'm', 'f' : iPoint := 5;
            'M', 'F' : iPoint := 12;
          end;
          inc(arrPoints[D], iPoint);
        end;
      end; //for A
    end; //for D

  //Sort the arrays according to points >> Any sorting method
  For A := 1 to 8-1 do
    For D := a+1 to 8 do
      IF arrPoints[A] < arrPoints[D]
      then
        begin //swop elements of both arrays
```

```
sTemp           := arrDisplay[D];
arrDisplay[D]   := arrDisplay[A];
arrDisplay[A]   := sTemp;

iTemp           := arrPoints[D];
arrPoints[D]   := arrPoints[A];
arrPoints[A]    := iTemp;
end;

//Display results
redQ3.Lines.Clear;
redQ3.Paragraph.TabCount := 2;
redQ3.Paragraph.Tab[0]   := 80;
redQ3.Paragraph.Tab[1]   := 150;
redQ3.Lines.Add('Medal winning displays:');
redQ3.Lines.Add('Medal' + #9 + 'Display' + #9 + 'Points');
for A := 1 to 3 do
  redQ3.Lines.Add(arrMedal[A] + #9 + arrDisplay[A] + #9 + IntToStr(arrPoints[A]));
end;

procedure TfrmQuestion3.mnuQuitClick(Sender: TObject);
begin
  Application.Terminate;
end;

end.
```

ANNEXURE G: SOLUTION FOR QUESTION 1: JAVA

```
//Solution for Question 1...
import java.io.*;
import java.sql.*;
import javax.swing.*;
import java.util.Scanner;

public class TestQuestion1_Memo
{
    public static void main (String[] args) throws SQLException,IOException
    {

        BufferedReader inKb = new BufferedReader (new InputStreamReader
(System.in));

        Zoo DB = new Zoo();
        System.out.println();

        char choice = ' ';
        do
        {
            System.out.println("\n\n      MENU");
            System.out.println();
            System.out.println("      Option A");
            System.out.println("      Option B");
            System.out.println("      Option C");
            System.out.println("      Option D");
            System.out.println("      Option E");
            System.out.println("      Option F");
            System.out.println("      Option G");
            System.out.println();
            System.out.println("      Q - QUIT");
            System.out.println(" ");
            System.out.print("      Your choice? ");
            choice = inKb.readLine().toUpperCase().charAt(0);
            System.out.println(" ");
            String sql = "";
            switch(choice)
            {
                case 'A':          // Question 1.1
                    {
                        sql = "SELECT * FROM tblCarnivores ORDER BY FamilyName,
ScientificName";
                        DB.query(sql);
                        break;
                    }
//=====
                case 'B':          // Question 1.2
                    {
                        System.out.println("Enter the family name, e.g. Canidae");
                        String sX = inKb.readLine();
                        sql = "SELECT ScientificName, GeneralName,
EnclosureNo,EnclosureSize FROM tblCarnivores WHERE EnclosureNo LIKE 'ZE%' AND
FamilyName = '" + sX + "'";

                        DB.query(sql);
                        break;
                    }
//=====
                case 'C':          // Question 1.3
                    {
```

```

        sql = "SELECT Endangered, Count(NumAdults+NumYoung) AS
CountAnimals FROM tblCarnivores GROUP BY Endangered";
        DB.query(sql);
        break;
    }
//=====
    case 'D':        // Question 1.4
    {
        sql = "SELECT EnclosureNo,
Format(EnclosureSize/(NumAdults+NumYoung), '#.0#') AS SpacePerAnimal FROM
tblCarnivores WHERE GeneralName LIKE '%mongoose%'";
        DB.query(sql);
        break;
    }
//=====
    case 'E':        // Question 1.5
    {
        sql = "UPDATE tblCarnivores SET NumYoung = NumYoung + 3
WHERE EnclosureNo = 'ZF1'";

        DB.query(sql);
        break;
    }
//=====
    case 'F':        // Question 1.6
    {
        System.out.println("Enter the day of the month when the
first visit took place (for example 23)");
        String sX = inKb.readLine();
        sql = "SELECT tblCarnivores.EnclosureNo, GeneralName,
VisitDate, ReasonForVisit, Animal_ID FROM tblCarnivores, tblVetVisits WHERE
tblCarnivores.EnclosureNo = tblVetVisits.EnclosureNo AND Day(VisitDate) = "+
sX;

        DB.query(sql);
        break;
    }
//=====
    case 'G':        // Question 1.7
    {
        sql = "INSERT INTO tblVetVisits
(VisitDate,EnclosureNo,ReasonForVisit,FollowUp,Animal_ID) VALUES (#2012/10/25#,
'ZD5', 'Ear infection', true,'ZD5_3')";

        DB.query(sql);
        break;
    }
}
}while (choice != 'Q');

DB.disconnect();
System.out.println("Done");
}
}

```

ANNEXURE H: SOLUTION FOR QUESTION 2: JAVA**QUEST2 CLASS UNIT**

```
//Solution for Question 2 class unit...
public class Quest2Memo
{
    private String type;
    private int number;
    private double size;
    private char cat;

    public Quest2Memo(String type, int number, double size, char cat)
    {
        this.type = type;
        this.number = number;
        this.size = size;
        this.cat = cat;
    }

    public boolean isSuitable(char cat, int number)
    {
        boolean suitable = false;
        double space = 0;
        if (type.equalsIgnoreCase("XXX"))
        {
            space = size / number;

            if (cat == 'L' && space >= 18)
                suitable = true;

            if (cat == 'M' && (space >= 12 && space < 18))
                suitable = true;

            if (cat == 'S' && (space >= 7 && space < 12))
                suitable = true;
        }
        return suitable;
    }

    public String toString()
    {
        return type + "... " + cat + "\nEnclosure size: " + size + "\nNumber
of animals: " + number + "\n\n";
    }
    public void setAType(String type) {
        this.type = type;
    }

    public void setNumber(int number) {
        this.number = number;
    }

    public void setSize(double size) {
        this.size = size;
    }

    public void setCat(char cat) {
        this.cat = cat;
    }

    public String getAType() {
        return type;
    }
}
```

```

    }

    public int getNumber() {
        return number;
    }

    public double getSize() {
        return size;
    }

    public char getCat() {
        return cat;
    }
}

```

TEST CLASS (DRIVER CLASS)

```

//Solution for Question 2 Test class...
import java.io.*;
import java.util.Scanner;

public class TestQuestion2_Memo {

    static Quest2Memo[] enclosures = new Quest2Memo[30];
    static int cnt;

    public static void main(String[] args) throws Exception
    {
        BufferedReader kb = new BufferedReader(new
InputStreamReader(System.in));
        readFile();

        char choice = ' ';
        do {
            System.out.println("    MENU");
            System.out.println();
            System.out.println(" Option A");
            System.out.println(" Option B");
            System.out.println();
            System.out.println(" Q - QUIT");
            System.out.println("\n Your choice? ");
            choice =kb.readLine().charAt(0);
            switch (choice) {
                case 'A':
                    display();
                    break;
                case 'B':
                    optionB();
                    break;
                case 'Q':
                    System.out.println("Quit");
            }
        } while (choice != 'Q');
    }

    public static void readFile() {
        try
        {
            cnt = 0;
            Scanner sc = new Scanner (new FileReader("DataQ2.txt"));
            while (sc.hasNext())
            {

```

```

        String line = sc.nextLine();
        int pos1 = line.indexOf('; ', 0);
        String aType = line.substring(0, pos1);

        int posHash = line.indexOf('#', pos1);
        int numberAn = Integer.parseInt(line.substring(pos1 +
1, posHash));

        int pos2 = line.indexOf('; ', posHash);
        double size = Double.parseDouble(line.substring(posHash +
1, pos2));

        int posHash2 = line.indexOf('#', pos2);
        char cat = line.charAt(posHash2-1);

        enclosures[cnt] = new Quest2Memo(aType, numberAn, size, cat);
        cnt++;
    }
    sc.close();
}
catch (FileNotFoundException e) {
    System.out.println("File does not exist");
    System.exit(0);
}
catch (Exception f) {
    System.out.println(f);
}
}

public static void display() {
    System.out.println("List of all enclosures");
    System.out.println("=====");

    for (int k = 0; k < cnt; k++) {
        System.out.println("Enclosure number: " + (k+1)+"\n" +
enclosures[k].toString());
    }
}

private static void optionB() throws Exception {
    BufferedReader kb = new BufferedReader(new
InputStreamReader(System.in));

    boolean found = false;
    int count = 0;

    System.out.println("Enter the type of animal (for example Tiger)");
    String animal = kb.readLine();
    System.out.println("Enter the number of animals, e.g. 2");
    int numA = Integer.parseInt(kb.readLine());
    System.out.println("Enter the category (L/M/S)");
    char cat = kb.readLine().charAt(0);

    while (found == false && count < cnt)
    {
        if (enclosures[count].isSuitable(cat, numA))
        {
            found = true;
            enclosures[count].setAType(animal);
            enclosures[count].setNumber(numA);
            enclosures[count].setCat(cat);
        }
        else

```



```
        count++;
    }
    if (found == false)
    {
        System.out.println("No suitable enclosure was found");
    }
    else
    {
        System.out.println("\n\nThese animals were placed in enclosure
number " + (count +1 ));
        System.out.println("\n");
        display();
    }
}
}
```

ANNEXURE I: SOLUTION FOR QUESTION 3 WITH OOP: JAVA

```
import java.io.IOException;

public class TestQuestion3_Memo
{
    public static void main(String[] args) throws IOException
    {
        Question3_Memo test = new Question3_Memo();
        test.displayMenu();
    }
}

// Object class describing a Ticket object
public class Ticket
{
    private String section;
    private String display;
    private String gender;
    private int number;

    public Ticket(String ticket)
    {
        section = ticket.substring(0,1);
        display = ticket.substring(1,2);
        gender = ticket.substring(2,3);
        number = Integer.parseInt(ticket.substring(3,4));
    }

    public boolean isValid()
    {
        boolean valid = true;

        if (("ABCD".indexOf(display.charAt(0)) < 0) || (!(section.equals("A")) &&
        (!(section.equals("R")))) || ("MmFf".indexOf(gender.charAt(0)) < 0 ))
            valid = false;
        return valid;
    }

    public int getPointvalue()
    {
        if (gender.equals("M") || gender.equals("F") )
            return 12;
        else
            return 5;
    }

    public String getSection()
    {
        return section;
    }

    public String getDisplay()
    {
        return display;
    }

    public String getGender()
    {
        return gender;
    }
}
```

```

public int getNumber()
{
    return number;
}
}
//=====

import java.io.IOException;
import java.io.BufferedReader;
import java.io.InputStreamReader;

public class Question3_Memo
{
    String[] arrTic = {"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"}; // size 32

    String[] arrDisplay = {"AA", "AB", "AC", "AD", "RA", "RB", "RC", "RD"};
    int[] arrPoints = {0,0,0,0,0,0,0,0};
    String[] arrMedal = {"Gold", "Silver", "Bronze"};

    BufferedReader inKb;
//=====
// Option A
    public void validate()
    {
        System.out.println("Invalid entries");
        for (int count = 0; count < 32; count++) {

            Ticket ticket = new Ticket(arrTic[count]);
            if (ticket.isValid() == false) {

                System.out.println(arrTic[count]);
                arrTic[count] = "Z";
            }
        }
    }
//=====
//Option B
    public void getWinningNumber()
    {
        boolean valid = false;
        int win = (int)(Math.random() * 32);
        while (valid == false)
        {
            if(arrTic[win].equals("Z"))
            {
                win = (int)(Math.random() * 32);
                System.out.println("Invalid");
            }
            else
                valid = true;
        }
        System.out.println("The position of the winning ticket in the array: "
+ (win+1));
        System.out.println("The winning ticket: " + arrTic[win]);
    }
}

```

```
//=====
// Option C
// Identify Medal Winners
public void getMedalWinners() throws IOException
{
    //For each display add points:
    for (int d = 0; d < 8; d++) {
        for (int t = 0; t < 32; t++) {
            if( !(arrTic[t].equalsIgnoreCase("Z"))){

                Ticket ticket = new Ticket(arrTic[t]);

                String displayChoice = ticket.getSection() +
                    ticket.getDisplay();

                if (displayChoice.equalsIgnoreCase(arrDisplay[d])) {
                    arrPoints[d]= arrPoints[d] + ticket.getPointvalue();
                } //if
            } // if not Z
        } // for ticket
    } // for d

    //Sort the two arrays
    for (int a = 0; a < 8 -1; a++) {
        for (int d = (a+1); d < 8; d++) {
            if ( arrPoints[a] < arrPoints[d]){

                String tempD = arrDisplay[a];
                arrDisplay[a]= arrDisplay[d];
                arrDisplay[d] = tempD;

                int tempP = arrPoints[a];
                arrPoints[a] = arrPoints[d];
                arrPoints[d] = tempP;
            } // if
        } // for d
    } // for a

    //display medals
    System.out.println("Medal winning displays:");
    System.out.printf("%s%20s%20s\n", "Medal", "Display", "Points");
    for (int a = 0; a < 3; a++) {
        System.out.printf("%-8s%12s%21d\n", arrMedal[a],
            arrDisplay[a],arrPoints[a]);
    }
} // getMedalWinners

public void displayMenu() throws IOException
{
    inKb = new BufferedReader (new InputStreamReader (System.in));
    System.out.println();

    char choice = ' ';
    do
    {
        System.out.println("\n\n          MENU");
        System.out.println();
        System.out.println("          Option A");
        System.out.println("          Option B");
        System.out.println("          Option C");
        System.out.println();
        System.out.println("          Q - QUIT");
        System.out.println(" ");
    }
}

```

```
System.out.print("    Your choice? ");
choice = inKb.readLine().toUpperCase().charAt(0);
System.out.println(" ");
String sql = "";
switch(choice)
{
    case 'A':
        validate();
        break;
    case 'B':
        getWinningNumber();
        break;
    case 'C':
        getMedalWinners();
        break;
    case 'Q':
        System.out.println("QUIT");
        break;
}
}while(choice != 'Q');
}
}
```

ANNEXURE J: SOLUTION FOR QUESTION 3 WITHOUT OOP: JAVA

```

import java.io.IOException;
import java.io.BufferedReader;
import java.io.InputStreamReader;

public class Question3_Memo
{
    String[] arrTic = {"RCm158", "ADm33", "RCf250", "RAF5",
        "RRM32", "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"};

    //arrays used in solution for Option C
    String[] arrDisplay = {"AA", "AB", "AC", "AD", "RA", "RB", "RC", "RD"};
    int[] arrPoints = {0,0,0,0,0,0,0,0};
    String[] arrMedal = {"Gold", "Silver", "Bronze"};

    BufferedReader inKb;

    //=====
    // Option A
    public void validate()
    {
        System.out.println("Invalid entries");
        for (int c = 0; c < 32; c++)
        {
            char firstchar = arrTic[c].charAt(0);
            char secondchar = arrTic[c].charAt(1);
            // or String secondchar = arrTic[c].substring(1,2);
            char thirdchar = arrTic[c].charAt(2);

            if ("ABCD".indexOf(secondchar) < 0 || (firstchar != 'A' && firstchar
            != 'R' ) || ("MmFf".indexOf(thirdchar) < 0 ))
            {
                System.out.println(arrTic[c]);
                arrTic[c] = "Z";
            }
        }
    }
    //=====
    //Option B
    public void getWinningNumber()
    {
        boolean valid = false;
        int win = (int)(Math.random() * 32);
        while (valid == false)
        {
            if(arrTic[win].equals("Z"))
            {
                win = (int)(Math.random() * 32);
                System.out.println("Invalid");
            }
            else
                valid = true;
        }
        System.out.println("The position of the winning ticket in the array: "
        + (win+1));
        System.out.println("The winning ticket: " + arrTic[win]);
    }
}

```

}

```
//=====
// Option C
// Identify Medal Winners
public void getMedalWinners() throws IOException
{
    // Write code for Option C
    //For each display add points:
    for (int d = 0; d < 8; d ++){
        for (int t = 0; t < 32; t++){
            if( !(arrTic[t].equalsIgnoreCase("Z"))){
                String displayChoice = arrTic[t].substring(0,2);
                if (displayChoice.equalsIgnoreCase(arrDisplay[d])) {
                    char gender = arrTic[t].charAt(2);
                    if (gender == 'f' || gender == 'm')
                        arrPoints[d]= arrPoints[d] + 5;
                    else
                        arrPoints[d] = arrPoints[d]+ 12;
                } //if
            } // if not Z
        } // for t
    } // for d

    //Sort the two arrays
    for (int a = 0; a < 8 -1; a++){
        for (int d = (a+1); d < 8; d++){
            if ( arrPoints[a] < arrPoints[d]){

                String tempD = arrDisplay[a];
                arrDisplay[a]= arrDisplay[d];
                arrDisplay[d] = tempD;

                int tempP = arrPoints[a];
                arrPoints[a] = arrPoints[d];
                arrPoints[d] = tempP;
            } // if
        } // for d
    } // for a
    //display medals
    System.out.println("Medal winning displays:");
    System.out.printf("%s%20s%20s\n", "Medal", "Display", "Points");
    for (int a = 0; a < 3; a ++){
        System.out.printf("%-8s%12s%21d\n", arrMedal[a], arrDisplay[a],
            arrPoints[a]);
    }
} // getMedalWinners

public Question3_Memo() throws IOException
{
    inKb = new BufferedReader (new InputStreamReader (System.in));

    System.out.println();

    char choice = ' ';
    do
    {
        System.out.println("\n\n          MENU");
        System.out.println();
        System.out.println("          Option A");
        System.out.println("          Option B");
        System.out.println("          Option C");
        System.out.println();
    }
}

```

```
System.out.println("    Q - QUIT");
System.out.println(" ");
System.out.print("    Your choice? ");
choice = inKb.readLine().toUpperCase().charAt(0);
System.out.println(" ");
String sql = "";
switch(choice)
{
    case 'A':
        validate();
        break;
    case 'B':
        getWinningNumber();
        break;
    case 'C':
        getMedalWinners();
        break;
    case 'Q':
        System.out.println("QUIT");
        break;
}
}while(choice != 'Q');
}

public static void main(String[] args) throws IOException
{
    new Question3_Memo();
}
}
```