basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

NOVEMBER 2010

MEMORANDUM

MARKS: 120

This memorandum consists of 26 pages.

GENERAL INFORMATION:

- **Pages 2 – 11 contain the Delphi memoranda of possible solutions for QUESTIONS 1 to 3 in programming code.**
- **Pages 12 – 20 contain the Java memoranda of possible solutions for QUESTIONS 1 to 3 in programming code.**
- **Pages 21 – 26 contain Addenda A to F which includes a marking grid for each question for candidates using either one of the two programming languages.**
  Copies of the appropriate Addenda should be made for each learner to be completed during the marking session.

SECTION A: DELPHI

QUESTION 1: PROGRAMMING AND DATABASE

```
unit Question1_U;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, DB, ADODB, Grids, DBGrids, ExtCtrls, Buttons;

type
  TfrmSightings = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    btnB: TButton;
    btnE: TButton;
    btnA: TButton;
    btnD: TButton;
    btnF: TButton;
    BitBtn1: TBitBtn;
    btnC: TButton;
    qrySightings: TADOQuery;
    tblSightings: TDataSource;
    grdSightings: TDBGrid;
    btnG: TButton;
    procedure btnBClick(Sender: TObject);
    procedure btnEClick(Sender: TObject);
    procedure btnDClick(Sender: TObject);
    procedure btnCClick(Sender: TObject);
    procedure btnFClick(Sender: TObject);
    procedure btnAClick(Sender: TObject);
    procedure btnGClick(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmSightings: TfrmSightings;

implementation

{$R *.dfm}
```

```
procedure TfrmSightings.btnAClick(Sender: TObject);          // Question 1.1
begin
  qrySightings.Active := False;
  qrySightings.SQL.Text := 'SELECT * ✓FROM tblSightings✓ ORDER BY✓ SightingID
DESC✓';
  qrySightings.ExecSQL;
  qrySightings.Open;
end;                                                                    (4)
//=========================================================================

procedure TfrmSightings.btnBClick(Sender: TObject);          // Question 1.2
begin
  qrySightings.Active := False;
  qrySightings.SQL.Text := 'SELECT DISTINCT ✓ Animal✓ FROM tblSightings✓ WHERE Young
= true✓';
  qrySightings.ExecSQL;
  qrySightings.Open;
end;                                                                    (4)
//=========================================================================

procedure TfrmSightings.btnCClick(Sender: TObject);          // Question 1.3
begin
  qrySightings.Active := False;
  qrySightings.SQL.Text := 'SELECT RangerID, Name, Surname, ✓ year(Now())✓ – ✓ year
(DateAppointed)✓ AS [TotalYears]✓ FROM tblRangers'✓;
  qrySightings.ExecSQL;
  qrySightings.Open;
end;                                                                    (6)
//=========================================================================

procedure TfrmSightings.btnDClick(Sender: TObject);          // Question 1.4
begin
  qrySightings.Active := False;
  qrySightings.SQL.Text := 'SELECT Animal✓,format(Avg(NumAnimals)✓,"0.00"✓)  AS
[AvgSighted]✓ FROM tblSightings✓ GROUP BY Animal✓';
  qrySightings.ExecSQL;
  qrySightings.Open;
end;                                                                    (6)
//=========================================================================
```

OR round(Avg(NumAnimals), 2)

```
procedure TfrmSightings.btnEClick(Sender: TObject);          // Question 1.5
var
  id : integer;
  numRecords : integer;
begin
  id := StrToInt(InputBox('Delete Sighting', 'Enter the ID of the sighting to
delete', '1')); ✓
  qrySightings.Active := False;
  qrySightings.SQL.Text := 'DELETE✓ FROM tblSightings✓ WHERE SightingID = '+
IntToStr(id) ✓;
 numRecords := qrySightings.ExecSQL;
  MessageDlg (IntToStr(numRecords) + ' record deleted.',mtInformation,[mbok],0);
end;                                                                    (4)
//=========================================================================
```

```
procedure TfrmSightings.btnFClick(Sender: TObject);          // Question 1.6
var
  numRecords : integer;
begin
  qrySightings.Active := False;
  qrySightings.SQL.Text := 'UPDATE✓ tblSightings✓ SET✓ Animal = "White Rhino"✓ WHERE
Animal = "Rhino"✓';
  numRecords := qrySightings.ExecSQL;
  MessageDlg (IntToStr(numRecords) + ' records updated.', mtInformation,[mbok],0);
end;                                                                          (5)
//=============================================================================

procedure TfrmSightings.btnGClick(Sender: TObject);          // Question 1.7
begin
  qrySightings.Active := False;
  qrySightings.SQL.Text := 'SELECT SightingDate,Name,Surname✓ FROM tblSightings,
tblRangers✓ WHERE tblSightings.RangerID✓ = tblRangers.RangerID✓ AND Animal =
"Elephant"✓ AND SightingDate > #30/04/2010#'; ✓
  qrySightings.ExecSQL;
  qrySightings.Open;
end;                                                                          (6)
                                                                             [35]
end.
```

QUESTION 2: OBJECT-ORIENTED PROGRAMMING

unit uCompetitorXXXX;

```
interface

uses SysUtils;
```

***//* Q 2.1.1**

```
type TCompetitor = class
   private ✓                (3)
    name : String; ✓
    largeGameCount : integer;   }
    smallGameCount : integer;   }    ✓
    birdCount : integer;

   public
    constructor Create; overload;
    constructor Create(sName : String); overload;
    function toString : String;
    function totalAnimals : integer;
    procedure spotLarge;
    procedure spotSmall;
    procedure spotBird;
    function calculatePoints : integer;
    function getName : String;
    function mostSpotted : String;
end;

implementation

{ Competitor }
//============================================================================
constructor TCompetitor.Create;
begin

end;
```

Q 2.1.1
(1) Four private variables
(1) Declare name as private string
(1) All count variables declared as private integers

***//* Q 2.1.2** **(3)**

```
constructor TCompetitor.Create(sName : String);   ✓
begin
  name := sName;
  largeGameCount := 0;
  smallGameCount := 0;      ✓
  birdCount := 0;
end;

procedure TCompetitor.spotLarge;
begin
   inc(largeGameCount);
end;

procedure TCompetitor.spotSmall;
begin
  inc(smallGameCount);          ✓
end;

procedure TCompetitor.spotBird;
begin
   inc(birdCount);
end;
//============================================================================
```

Q 2.1.2
(a) (1) Receive name as String
(1) Initialize all instance fields
(b) (1) Remove comment-signs from the code in the three given 'spot'-methods

Q 2.1.3
(3) Multiply each category with the correct value add the values and assign to Result
Subtract 1 mark for each type of error, not for the same type of error

```
// Q 2.1.3                          (3)
function TCompetitor.calculatePoints: integer;
begin
   Result := largeGameCount * 5 + smallGameCount * 3 + birdCount * 2 ✓✓✓;
end;
//=============================================================
```

Q 2.1.4
(1) Return type integer
(1) Return sum of animal counts

```
// Q 2.1.4                          (2)
function TCompetitor.totalAnimals: integer✓;
begin
  Result := largeGameCount + smallGameCount + birdCount; ✓
end;

//=============================================================================
```

Q 2.1.5
(1) Return type String
(1) Return name

```
// Q 2.1.5                          (2)
function TCompetitor.getName: String; ✓
begin
  Result := name ✓
end;

//=============================================================
```

Q 2.1.6
(1) Return type String
(1) If to get Large Game
(1) If to get Small game
(1) Get Birds

```
// Q 2.1.6                          (4)
function TCompetitor.mostSpotted: String✓;
begin
  if (largeGameCount > smallGameCount) AND (largeGameCount > birdCount) ✓ then
    Result := 'Large Game'
  else if (largeGameCount < smallGameCount) AND (smallGameCount > birdCount) ✓ then
    Result := 'Small Game'
  else ✓                           // Small game AND birds must have else statements, -
                                   // otherwise another if statement for birds
    Result := 'Bird';

end;                               // Correct variations of this code must be accepted
//=============================================================================
```

Q 2.1.7
(1) New line (#13)
(1) Add strings
(1) All count elements added
(1) Last line added
(1)Tab (#9) added

```
//Q 2.1.7                          (5)
function TCompetitor.toString: String;
var
  objStr : String;
begin

  objStr := 'Competitor : ' + name + #13✓;
  objStr := objStr✓ + 'Large : ' + IntToStr(largeGameCount) + '  Small : '
+ IntToStr(smallGameCount) + '  Bird : ' + IntToStr(birdCount) + #13✓;
  objStr := objStr + 'Total Animals : ' + #9✓ + IntToStr(totalAnimals) ✓;
  Result := objStr;
end;               // May use more tabs (#9) than indicated here
//=============================================================================
```

unit QuestTwoXXXX_U;

```
interface

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

type
  TfrmQuest2 = class(TForm)
    redOutput: TRichEdit;
```

```
    MainMenu1: TMainMenu;
    mnuAQuestion2: TMenuItem;
    mnuBQuestion2: TMenuItem;
    Quit1: TMenuItem;
    procedure Quit1Click(Sender: TObject);
    procedure mnuAQuestion2Click(Sender: TObject);
    procedure mnuBQuestion2Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmQuest2: TfrmQuest2;

implementation

{$R *.dfm}
```

//Q 2.2.1 (20)

```
uses
  uCompetitorXXXX; ✓
var
  Competitor : TCompetitor;  ✓

procedure TfrmQuest2.Quit1Click(Sender: TObject);
begin
  Application.Terminate;
end;

procedure TfrmQuest2.FormActivate(Sender: TObject);
var
  tFile : textfile;
  sName, sAnimal, sAnimalName : String;
  iValid, iInvalid, iBracket : integer;
  sLetter :char;
begin

  redOutput.Lines.Clear;
  iValid := 0;
  iInvalid := 0;  ✓

  If FileExists('Sightings.txt') <> TRUE then ✓
    begin
      ShowMessage('File not found');  ✓
      Application.Terminate;
    end
    AssignFile(tFile, 'Sightings.txt');   ✓
    Reset(tFile); ✓
    ReadLn(tFile, sName); ✓
    Competitor := TCompetitor.Create✓(sName)✓;
    while NOT EOF(tFile) DO  ✓
        begin
         ReadLn(tFile, sAnimal);  ✓
         iBracket := pos('(', sAnimal);
         sAnimalName := copy(sAnimal,1,iBracket - 1); ✓
         sLetter := sAnimal[iBracket + 1]; ✓
```

Q 2.2.1
(1) uses object class
(1) Declare object variable
(1) Initialise two counter variables
(1) Test if file exists
(1) If not, display message & terminate
(1) Assign file
(1) Open file for reading
(1) Read name of competitor from file
(2) Create competitor object
(1) Loop with begin & end in correct places
(1) Read line from text file
(1) Copy animal name from line
(1) Copy animal category from line
(1) Case (OR if-statements)
for each valid category
(1) call method to inc counter
(1) add 1 to valid counter
(1) For invalid category
(1)inc counter & display message
(1) Display the number of valid & invalid categories processed

```
        case sLetter of  ✓
          'L' : begin
                  Competitor.spotLarge;
                  inc(iValid);
                end;
          'S' : begin
                  Competitor.spotSmall;      ✓✓✓
                  inc(iValid);
                end;
          'B' : begin
                  Competitor.spotBird;
                  inc(iValid);
                end;
          else begin
                  redOutput.Lines.Add(sAnimal + ' is not a valid animal'); ✓
                  Inc(iInvalid);

                end;
        end; // case

      end;
    CloseFile(tFile);
    redOutput.Lines.Add('');
    redOutput.Lines.Add(IntToStr(iValid) + ' valid categories processed'); ✓
    redOutput.Lines.Add(IntToStr(iInvalid) + ' invalid categories
                                                        processed');
end;
//============================================================================
```

***//* Q 2.2.2 (2)**

```
procedure TfrmQuest2.mnuAQuestion2Click(Sender: TObject);
begin
   redOutput.Lines.Clear;
   redOutput.Lines.Add(Competitor.toString);  ✓✓
end;
//============================================================================
```

Q 2.2.2
(1) Call the toString method of the object
(1) in a display statement

***//* Q 2.2.3 (5)**

```
procedure TfrmQuest2.mnuBQuestion2Click(Sender: TObject);
var
  tFile : textfile;
  fileName: String;
begin
  fileName := Competitor.getName + '.txt';  ✓
  AssignFile(tFile, fileName);
  Rewrite(tFile);                           ✓

  Writeln(tFile, 'Competitor : ' + Competitor.getName);
  WriteLn(tFile, 'Total Animals : ' + IntToStr(Competitor.totalAnimals));  ✓✓
  WriteLn(tFile, 'Points : ' + IntToStr(Competitor.calculatePoints));
  WriteLn(tFile, 'Most Sighted Category : ' + Competitor.mostSpotted);

  CloseFile(tFile);  ✓

  redOutput.Lines.Clear;
  redOutput.Lines.Add('Results Successfully Written to File');
end;
end.
//============================================================================
```

Q 2.2.3
(1) Construct the name of the new file correctly
(1) Open a new file
(1) Construct the information to write to the file correctly
(1) Write to the file
(1) Close the file

[49]

QUESTION 3: DELPHI PROGRAMMING

NB: This is only a sample – learners may answer this question in any way they see fit. Make use of the generalized rubric in the mark sheets for marking.

```
unit Question3_U;

interface

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

type
  TfrmQuestion3 = class(TForm)
    redOutput: TRichEdit;
    pnlButtons: TPanel;
    btnA: TButton;
    btnB: TButton;
    BitBtn1: TBitBtn;
    procedure FormCreate(Sender: TObject);
    procedure btnAClick(Sender: TObject);
    procedure btnBClick(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmQuestion3: TfrmQuestion3;

implementation

{$R *.dfm}
```

//Question 3.1 (3)

```
var
  arrEntries : array[1..12] ✓of string; ✓

procedure TfrmQuestion3.FormCreate(Sender: TObject);
 begin
  arrEntries[1]  := '12,15:02h00';
  arrEntries[2]  := '13,10:05h00';
  arrEntries[3]  := '9,20:06h00';
  arrEntries[4]  := '10,15:09h00';
  arrEntries[5]  := '7,8:10h00';
  arrEntries[6]  := '10,10:11h00';
  arrEntries[7]  := '12,18:14h00';
  arrEntries[8]  := '7,18:17h00';
  arrEntries[9]  := '11,7:19h00';
  arrEntries[10] := '10,10:20h00';
  arrEntries[11] := '2,1:23h00';
  arrEntries[12] := '12,17:24h00'; ✓
end;
//============================================================================
```

Q 3.1
(2) Declare array of strings
(1) Remove the comments-signs from given statements assigning strings to array

//Quest 3.2 (17)

```
procedure TfrmQuestion3.btnAClick(Sender: TObject);
var
   K, iDistance :integer;
   iWaterX, iWaterY, xPos, yPos   :integer;
   iComma, iColon :integer;
   sTime:string;
begin
 iWaterX := 10;  }      ✓
 iWaterY := 10;  }

 redOutput.Paragraph.TabCount := 4;
 redOutput.Paragraph.Tab[0] := 60;
 redOutput.Paragraph.Tab[1] := 100;
 redOutput.Paragraph.Tab[2] := 130;
 redOutput.Paragraph.Tab[2] := 150;

  redOutput.Lines.Add('Distances from the watering hole'); ✓
  redOutput.Lines.Add('Time' + #9 + 'Distance(km)  ' + #9 + 'X-pos   Y-pos'); ✓
  for K := 1 to 12 do✓
    begin
     iComma := pos(',', arrEntries[K]); ✓
     xPos := StrToInt(copy(arrEntries[K],1,iComma-1)); ✓
     iColon := pos(':', arrEntries[K]); ✓
     yPos := StrToInt(copy(arrEntries[K],iComma + 1,iColon - iComma-1)); ✓
     iDistance:=  Round✓ (Sqrt✓ (Sqr(xPos - iWaterX  ) ✓ + Sqr(yPos –
                                                    iWaterY ))); ✓
     delete(arrEntries[K],1,iColon); ✓
     sTime:= arrEntries[K]; ✓
     redOutput.Lines.Add(sTime ✓+ #9 + '       ' + IntToStr(iDistance) ✓ + #9
                 + #9+ IntToStr(xPos) + #9 + IntToStr(yPos)); ✓
   end;
end;

//=====================================================================
```

Q 3.2
(1) Initialise water x & y positions
(2) Display heading & subheadings
(1) Loop with begin & end in correct places
Inside loop:
(2) Extract x-pos,
(2) Extract y-pos,
(2) Extract time from array string
(3) Calculate distance,
(1) Round answer off,
(3) Display all information

//Question 3.3 (16)

```
procedure TfrmQuestion3.btnBClick(Sender: TObject);
var
  K, iNumber, iCount, iLength  :integer;
  iRandomNumber, L :integer;
  sAnimal, sTag :string;

begin
  redOutput.Paragraph.TabCount := 1;
  redOutput.Paragraph.Tab[0] := 70;
  Randomize;
  redOutput.Clear;
  iNumber := StrToInt(InputBox('Animal tags',
           'Enter the number of different types of
             animals in the group?', '')); ✓
  for K := 1 to iNumber do✓
    begin✓
      sAnimal := InputBox('Animal Tags', 'Enter the
            name of animal type ' + intToStr(K), '');✓
      iCount := StrToInt(InputBox('Animal Tags',
                 'Enter the number of animals of type '
               + intToStr(K) + ' in the group', ''));✓

      sTag := copy(sAnimal, 1, 2); ✓
```

Q 3.3
(1) Enter the number of different types of animals,
(1) Loop from 1 to the number of different types
(1) Inside loop:
(1) Enter animal type
(1) Enter the number in the group
(1) Extract first 2 letters and assign to tag
(2) Extract the last letter,
(2) Generate random number in correct range,
(1) Validate an even number,
(1) Add random number to tag, and hyphen (1) Heading
(1) Inner loop to add the number of animals in each group,
(2) Display the correct data aligned correctly inside inner loop

```
      iLength := length(sAnimal);
      sTag := sTag + copy(sAnimal, iLength✓,1);✓
      repeat
        iRandomNumber := Random(900) ✓ + 100; ✓
      until iRandomNumber mod 2 = 0; ✓
      redOutput.Lines.Add('');
      sTag := sTag + IntToStr(iRandomNumber) ✓;
      redOutput.Lines.Add(sAnimal + #9 + 'Tag number');✓
      for L := 1 to iCount do✓
        begin
         redOutput.Lines.Add(intToStr(L) + '.' + #9 + sTag ✓+ '-' +intToStr(L)); ✓
        end;
    end;
end;
//============================================================================
end.                                                                    [36]
```

```
OR
iRandomNumber:= Random(899)+ 100;
if iRandomNumber mod 2 <> 0 then
      inc(iRandomNumber);
```

END OF SECTION A: DELPHI **TOTAL SECTION A: 120**

SECTION B: JAVA

QUESTION 1: PROGRAMMING AND DATABASE

```
import java.io.*;
import java.sql.*;
import javax.swing.*;
import java.util.Scanner;

public class TestSightings
 {
  public static void main (String[] args) throws SQLException,IOException
   {
    BufferedReader inKb = new BufferedReader (new InputStreamReader(System.in));

    Sightings DB = new Sightings();
    System.out.println();
    char choice = ' ';
     do
      {
         System.out.println("      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 tblSightings✓ ORDER BY ✓SightingID DESC✓";
            DB.A(sql);
            break;
          }                                                                  (4)
//=========================================================================
         case 'B':                                          //Question 1.2
         {
            sql = "SELECT DISTINCT ✓ Animal✓ FROM tblSightings✓ WHERE Young =
                                                            true✓";
           DB.B(sql);
           break;
         }                                                                   (4)
//=========================================================================
        case 'C':                                           //Question 1.3
         {
            sql = "SELECT RangerID, Name, Surname, ✓ year(Now())✓ – ✓ year
            (DateAppointed)✓ AS [TotalYears]✓ FROM tblRangers"✓;
            DB.C(sql);
            break;
         }                                                               (6)
//=========================================================================
```

```
       case 'D':                                          //Question 1.4
          {
             sql = "SELECT Animal✓, format(Avg(NumAnimals) ✓, '0.00'✓) AS
                   [AvgSighted]✓ FROM tblSightings✓ GROUP BY Animal✓";

             DB.D(sql);              OR round(Avg(NumAnimals),2)
             break;
          }                                                              (6)
//=============================================================================
      case 'E':   //Question 1.5
          {
             Scanner kb = new Scanner (System.in);
             System.out.println();
             System.out.println("Enter the ID of the sighting to delete");
             int id = kb.nextInt();✓
             sql = "DELETE✓FROM tblSightings✓ WHERE SightingID = "+id+"✓";
             DB.E(sql);
             break;
          }                                                              (4)
//=============================================================================
          case 'F':                                          //Question 1.6
          {
             sql = "UPDATE✓ tblSightings✓ SET✓ Animal = 'White Rhino'✓ WHERE
                   Animal = 'Rhino'✓";
             DB.F(sql);
             break;
          }                                                              (5)
//=============================================================================
          case 'G':                                          //Question 1.7
           {
            sql = "SELECT SightingDate,Name,Surname✓ FROM tblSightings,
                   tblRangers✓ WHERE tblSightings.RangerID=✓ tblRangers.RangerID✓ AND
                   Animal = 'Elephant'✓ AND SightingDate > #30/04/2010#";✓

            DB.G(sql);
            break;
          }                                                              (6)
//=============================================================================

       } //switch
      }while (choice != 'Q');

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

[35]

QUESTION 2: OBJECT-ORIENTED PROGRAMMING

CompetitorXXXX.java

```
public class CompetitorXXXX
{
    // Q 2.1.1                          (3)
    private✓ String name; ✓
    private int largeGameCount;
    private int smallGameCount;          ✓
    private int birdCount;

        public CompetitorXXXX ()
    {

    }
    //=========================================================================
    // Q 2.1.2                          (3)
    public CompetitorXXXX (String aName) ✓
    {
        name = aName;

        largeGameCount = 0;
        smallGameCount = 0;              ✓
        birdCount = 0;
    }

    public void spotLarge()
    {
        largeGameCount++;
    }

    public void spotSmall()
    {
        smallGameCount++;                ✓
    }

    public void spotBird()
    {
        birdCount++;
    }

     //=========================================================================
    // Q 2.1.3                          (3)



    public int calculatePoints()
    {
      return largeGameCount * 5✓ + smallGameCount * 3 ✓+ birdCount * 2✓;
    }
//=============================================================================


    // Q 2.1.4                          (2)
    public int✓ totalAnimals()
    {
        return largeGameCount + smallGameCount + birdCount; ✓
    }
```

Q 2.1.1
(1) Four private variables
(1) Declare name as private string
(1) All count variables declared as private integers

Q 2.1.2
(1) Receive name as String
(1) Initialize all instance fields
(1) Remove comment-signs from code in the three given spot-methods

Q 2.1.3
(3) Multiply each category with the correct value, add the values and return value
Subtract 1 mark for each type of error, not for the same type of error

Q 2.1.4
(1) Return type integer
(1) Return sum of animal counts

```
    //============================================================================
    // Q 2.1.5                                  (2)
     public String getName()✓
     {
         return name; ✓
     }
     //============================================================================
```

Q 2.1.5
(1) Return type String
(1) Return name

Q 2.1.6
(1) Return type String
(1) If to get Large Game
(1) If to get Small game
(1) Get Birds

```
    // Q 2.1.6                                  (4)
     public String✓ mostSpotted()
     {
         if (largeGameCount > smallGameCount & largeGameCount > birdCount) ✓
         {

             return "Large Game";
         }
         else if (smallGameCount > largeGameCount & smallGameCount > birdCount)✓
         {
             return "Small Game";
         }
         else✓                  // Small game AND birds must have else statements, -
                                // otherwise another if statement for birds
         {
             return "Bird";
         }                              // Correct variations of this code must be accepted
     }
     //============================================================================
```

Q 2.1.7
(1) New line (\n)
(1) Add strings
(1) All count elements added
(1) Add last line
(1) Tab (\t) added

```
    //Q 2.1.7                                  (5)
     public String toString()
     {
         String objStr = "Name : " + name + "\n"✓;
         objStr = objStr ✓+ "Large : " + largeGameCount + "  Small : "
    + smallGameCount + "  Bird : " + birdCount + "\n"✓;
         objStr = objStr + "Total Animals: \t"✓ + totalAnimals();✓

         return objStr;
     }

}        // Accept correct use of formatter to construct the string

//============================================================================
```

TestCompetitorXXXX.java

```
import javax.swing.*;
import java.io.*;
import java.util.*;

public class TestCompetitorXXXX
{
```

// Q 2.2.1 (20)

```
public static void main(String args[]) throws Exception
{
 CompetitorXXXX Competitor = new CompetitorXXXX ();✓

 File f = new File("Sightings.txt");✓
 if (!f.exists())✓
 {
   System.out.println("File not found");
   System.exit(0);                                  } ✓
 }
    Scanner sc = new Scanner(f); ✓

    String sName = sc.nextLine()✓;
    Competitor✓  = new CompetitorXXXX ✓ (sName✓);

    int valid = 0;    } ✓
    int invalid = 0;  }

    while (sc.hasNextLine())✓
    {
      String sAnimal = sc.nextLine();✓
      int bracket = sAnimal.indexOf('(');
      String animal = sAnimal.substring(0,bracket); ✓
      char letter = sAnimal.charAt(bracket + 1); ✓

      switch (letter) ✓
      {
        case 'L' : Competitor.spotLarge();
                   valid++;
                   break;
        case 'S' : Competitor.spotSmall();
                   valid++;                          } ✓✓✓
                   break;
        case 'B' : Competitor.spotBird();
                   valid++;
                   break;
        default: System.out.println(animal + " is not
                 in a valid category");
                 invalid++;✓
                 break;
       }

 System.out.println();
 System.out.println(valid + " valid categories processed");     } ✓
 System.out.println(invalid + " invalid categories processed");}

 sc.close();
 }
//=============================================================================
```

Q 2.2.1

(2) Declare object variable
(1) Initialise two counter variables
(1) Test if file exists
(1) Display message & terminate
(1) Assign file
(1) Open file for reading
(1) Read name of competitor from file
(3) Create competitor object
(1) Loop with curly brackets in correct places
(1) Read line from text file
(1) Copy animal name from line
(1) Copy animal category from line
(1) switch OR if-statements
(1) For each valid category
(1) Call method to inc counter
(1) Add 1 to valid counter
For invalid category
(1) Inc counter & display message
(1) Display the number of valid & invalid categories processed

```
BufferedReader inKb = new BufferedReader (new InputStreamReader (System.in));
 char ch = ' ';
 while (ch != 'Q')
  {
      System.out.println();
      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(" ");
      System.out.print("       Your choice? ");
      ch = inKb.readLine().toUpperCase().charAt(0);

      switch (ch)
      {
    // Q 2.2.2                          (2)
        case 'A':
         {
           System.out.println();
           System.out.println(Competitor.toString());✓✓
           break;
         }
//=============================================================================
    // Q 2.2.3                          (5)
        case 'B':
        {
          PrintWriter fOut = new PrintWriter(new File✓ (Competitor.getName()+
                                                        ".txt"));✓

          fOut.println("Competitor : " + Competitor.getName());
          fOut.println("Total Animals : " + Competitor.totalAnimals());
          fOut.println("Points : " + Competitor.calculatePoints());
          fOut.println("Most Sighted Category : " + Competitor.mostSpotted());  ✓✓

          fOut.close();✓

          System.out.println();
          System.out.println("Results Successfully Written to File");
          break;
        }

        case 'Q':
        {
          System.exit(0);
        } // case

      } // switch

  } // while

 } // main

} // class
//=============================================================================
```

[49]

Q 2.2.2
(1) Call the toString method of the object
(1) in a display statement

Q 2.2.3
(1) Construct the name of the new file correctly
(1) Open a new file
(1) Construct the information to write to the file correctly
(1) Write to the file
(1) Close the file

QUESTION 3: JAVA PROGRAMMING

NB: This is only a sample – learners may answer this in any way they see fit. Make use of the generalized rubric in the mark sheets for marking.

TestDistances.java

//Question 3.1 (3)

```
import java.util.Scanner;
import java.io.*;
import javax.swing.*;
public class TestDistances
{
  String ✓ [] arrEntries = new String [12]; ✓
  public TestDistances()
   {
      arrEntries[0] =  "12,15:02h00 ";
      arrEntries[1] =  "13,10:05h00 ";
      arrEntries[2] =  "9,20:06h00 ";
      arrEntries[3] =  "10,15:09h00 ";
      arrEntries[4] =  "7,8:10h00 ";
      arrEntries[5] =  "10,10:11h0";
      arrEntries[6] =  "12,18:14h00";
      arrEntries[7] =  "7,18:17h00";
      arrEntries[8] =  "11,7:19h00";
      arrEntries[9] =  "10,10:20h00";
      arrEntries[10]=  "2,1:23h00";
      arrEntries[11]=  "12,17:24h00";✓
   }
//=====================================================================
```

Q 3.1
(2) Declare array of strings
(1) Uncomment given statement assigning string to array

//Question 3.2 (17)

```
public void displayDistances()
 {
   int waterX = 10;
   int waterY = 10;   ✓
   int xPos;
   int yPos;
   System.out.println("Distances from the
                          watering hole");✓
   System.out.printf("%-10s%-15s%-10s%-10s",
                        "Time","Distance(km)",
                        "X-pos","Y-pos");✓
   System.out.println();
   int distance;
   String time;
   for (int i=0;i<12;i++)✓
    {
      String line = arrEntries[i];
      int psnComma = line.indexOf(','); ✓
      xPos = Integer.parseInt(line.substring(0,psnComma)); ✓
      int psnColon = line.indexOf(':'); ✓
      yPos = Integer.parseInt(line.substring(psnComma +
                                1,psnColon)); ✓
      time = line.substring(psnColon+1); ✓✓



      distance =  (int)(Math.round✓(Math.sqrt✓ (Math.pow((xPos– waterX),2) ✓
```

Q 3.2
(1) Initialise waterX & waterY
(2) Display heading & subheadings
(1) Loop with curly brackets correctly placed
Inside loop:
(2) Extract x-pos,
(2) Extract y-pos,
(2) Extract time from array string
(3) Calculate distance,
(1) Round answer off,
(3) Display all information

```
                                + Math.pow((yPos - waterY ),2))); ✓


      System.out.printf("%-10s%-15s%-10s%-10s", time, ✓distance ✓,xPos,yPos); ✓
      System.out.println();

    }
 }
//=====================================================================
```

//Question 3.3 (16)

```
public void tags()
      {
System.out.print("Enter the number of different
                 types of animals in the group: ");
int n = inKb.nextInt();✓
for (int i = 1; i <= n; i++)✓
{✓
  inKb.nextLine();
  System.out.println();
  System.out.print("Enter animal type "+ i + ": ");
  String animal = inKb.nextline();✓

  System.out.print("Enter the number of this type of
                        animal in the group: ");
  int number = inKb.nextInt();✓

  String tag = animal.substring(0,2); ✓

  tag = tag + animal.charAt(animal.length()✓ - 1);✓
  int randomNum;
  do
  {
    randomNum = (int)(Math.random()✓ * 900 + 100); ✓
  }
  while (randomNum % 2 !=0); ✓


  tag = tag + randomNum; ✓
  System.out.printf("%-20s%-20s",animal,"Tag number");✓
  System.out.println();
  for (int j = 1; j<= number; j++)✓
   {
     System.out.printf("%-20s%-20s",j+".",tag✓+"-"+j✓);
     System.out.println();
   }
  }
}
//=====================================================================

public static void main(String[] args) throws Exception
 {
   Scanner input = new Scanner(System.in);
   TestDistances obj = new TestDistances();
   char ch = ' ';
```

Q 3.3
(1) Enter the number of different types of animal
(1) Loop from 1 to the number of different types
(1)Inside loop:
(1) Enter animal type
(1) Enter the number in the group
(1) Extract first 2 letters and assign to tag
(2) Extract the last letter
(2) Generate number in correct range
(1) Validate even number
(1) Add random number to tag
(1) Heading
(1) Inner loop to the number of animals in each group
(2) Display the correct data aligned correctly inside inner loop

```
OR
randomNum =(int)(Math.random()*899+100);
if (randomNum % 2 !=0)
   randomNum++;
```

```
   while (ch != 'Q')
   {
       System.out.println();
       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(" ");
       System.out.print("       Your choice? ");

       ch = input.nextLine().toUpperCase().charAt(0);

       switch (ch)
       {
          case 'A':
          {
            obj.displayDistances();
            break;
          }

          case 'B':
          {
            obj.tags();
            break;
          }

          case 'Q':
          {
             System.exit(0);
          } // case

        } // switch

      } // while
   }
 }
//===========================================================================

}
```

Alternative structure: not advised but must be accepted:
The code can be used in place of the case statement. See electronic solution.

[36]

END OF SECTION B: JAVA **TOTAL SECTION B: 120**

GRAND TOTAL: 120

ADDENDUM A

QUESTION 1: DELPHI – PROGRAMMING AND DATABASE

CENTRE NUMBER: ……	EXAMINATION NUMBER:……

QUESTION 1: DELPHI – MARKING GRID

QUESTION	ASPECT	MAX. MARKS	LEARNER'S MARKS
1.1	SELECT * ✓FROM tblSightings✓ ORDER BY ✓SightingID DESC✓	4	
1.2	SELECT DISTINCT ✓ Animal✓ FROM tblSightings✓ WHERE Young = true✓	4	
	SELECT DISTINCT ✓ Animal✓ FROM tblSightings✓ WHERE Young = Yes✓		
	SELECT DISTINCT ✓ Animal✓ FROM tblSightings✓ WHERE Young = On✓		
1.3	SELECT RangerID, Name, Surname, ✓ year(Now())✓ – ✓ year (DateAppointed)✓ AS [TotalYears]✓ FROM tblRangers	6	
	SELECT RangerID, Name, Surname, ✓ year(Now())✓ – ✓ year (DateAppointed)✓ AS TotalYears✓ FROM tblRangers		
	SELECT RangerID, Name, Surname, ✓ 2010 ✓ – ✓ year (DateAppointed)✓ AS [TotalYears]✓ FROM tblRangers		
	SELECT RangerID, Name, Surname, ✓ year(Date())✓ – ✓ year (DateAppointed)✓ AS [TotalYears]✓ FROM tblRangers		
1.4	SELECT Animal✓, format(Avg(NumAnimals) ✓, "0.00"✓) AS [AvgSighted]✓ FROM tblSightings✓ GROUP BY Animal✓	6	
	SELECT Animal✓, round(Avg(NumAnimals) ✓,2✓) AS AvgSighted✓ FROM tblSightings✓ GROUP BY Animal✓		
1.5	Input statement✓ DELETE✓FROM tblSightings✓ WHERE SightingID = +IntToStr(id)✓	4	
1.6	UPDATE✓ tblSightings✓ SET✓ Animal = "White Rhino"✓ WHERE Animal = "Rhino"✓	5	
	UPDATE✓ tblSightings✓ SET✓ Animal = "White Rhino"✓ WHERE Animal Like "%Rhino%"✓ **//OR** "%Rhino"		
1.7	SELECT SightingDate,Name,Surname✓ FROM tblSightings, tblRangers✓ WHERE tblSightings.RangerID = ✓ tblRangers.RangerID✓ AND Animal = "Elephant"✓ AND SightingDate > #30/04/2010#	6	
	SELECT SightingDate,Name,Surname✓ FROM tblSightings,tblRangers✓ WHERE tblSightings.RangerID=✓ tblRangers.RangerID✓ AND Animal = "Elephant"✓ AND month(SightingDate) > 4		
	SELECT SightingDate,Name,Surname✓ FROM tblSightings,tblRangers✓ WHERE tblSightings.RangerID=✓ tblRangers.RangerID✓ AND Animal Like "Elephant"✓ AND SightingDate > #30/04/2010#		
	TOTAL:	35	

ADDENDUM B

QUESTION 2 - DELPHI: OBJECT-ORIENTED PROGRAMMING

CENTRE NUMBER:……	EXAMINATION NUMBER: ……

QUESTION 2 DELPHI – MARKING GRID

QUESTION	ASPECT	MAX. MARKS	LEARNER'S MARKS
2.1			
2.1.1	**Define attributes for TCompetitor**: four private (1) fields, three integers named correctly (1) and one string (1)	**3**	
2.1.2	**Constructor:** name parameter (1) initialise all instance fields (1) remove the comment-signs from code in the 'spot'-methods (1)	**3**	
2.1.3	**calculatePoints:** (3) multiply counters by correct values and add results	**3**	
2.1.4	**totalAnimals:** returns integer (1) containing sum of counters (1)	**2**	
2.1.5	**getName function:** Correct name and return type (1), Correct field returned (1)	**2**	
2.1.6	**mostSpotted function:** returns String (1) two if .. with else-statement each to determine highest count (3)	**4**	
2.1.7	**toString:** name and new line (1) and appends string (1) with category-points (1) add last line(1) with tab (1)	**5**	
2.2			
2.2.1	Declare a single TCompetitor object (2) Initialise counters to zero (1) If file does not exist (1)display message and exit (1) if file exists then assignfile (1) open file to read from(1) Read first line outside of loop into name variable (1) Call constructor method of TCompetitor (1) using name as a parameter (1) loop with begin and end correctly placed (1) Inside loop: Read line from file (1), get animal name (1) and category- letter (1) from string. Use letter and compare result (1) call appropriate method to increase count based on result (3) to handle all valid categories. Else increase invalid counter and display animal name in invalid category (1). Display number of valid and invalid entries (1)	**20**	
2.2.2	Call toString method of TCompetitor object (1) to display information(1)	**2**	
2.2.3	Use name of competitor to construct filename (1) ready the file for writing to new file (1) Call methods to construct output (1) and write to file (1), close the file (1)	**5**	
	TOTAL:	**49**	

ADDENDUM C

QUESTION 3: DELPHI PROGRAMMING

CENTRE NUMBER: ……	EXAMINATION NUMBER: ……

QUESTION 3 DELPHI – MARKING GRID

QUESTION	ASPECT	MAX. MARKS	LEARNER'S MARKS
3.1	(2) Declare arrEntries array (1) Remove comment-signs from statements assigning strings to array	**3**	
3.2	**Option A** (1) Assign 10 to waterX and waterY (2) Heading and subheadings (1) Loop through given array Inside Loop: (2) Extract xPos and (2) yPos (2) Extract time (3) Calculate distance, (1) Round down (3) Display info in loop	**17**	
3.3	**Option B** (1) Enter the number of different types of animals (1) Loop from 1 to the number of different types (1) Inside loop: (1) Enter animal type (1) Enter the number in the group (1) Extract first 2 letters and assign to tag (2) Extract the last letter (2) Generate number in correct range (1) Validate even number (1) Add random number and hyphen to tag (1) Heading (1) Inner loop to add the unique number of animals in each group (2) Display the correct data aligned correctly inside inner loop	**16**	
	TOTAL:	**36**	

ADDENDUM D

QUESTION 1: JAVA – PROGRAMMING AND DATABASE

CENTRE NUMBER: ……	EXAMINATION NUMBER:……

QUESTION 1: JAVA – MARKING GRID

QUESTION	ASPECT	MAX. MARKS	LEARNER'S MARKS
1.1	SELECT * ✓FROM tblSightings✓ ORDER BY ✓SightingID DESC✓	4	
1.2	SELECT DISTINCT ✓ Animal✓ FROM tblSightings✓ WHERE Young = true✓	4	
	SELECT DISTINCT ✓ Animal✓ FROM tblSightings✓ WHERE Young = Yes✓		
	SELECT DISTINCT ✓ Animal✓ FROM tblSightings✓ WHERE Young = 1✓		
1.3	SELECT RangerID, Name, Surname, ✓ year(Now())✓ – ✓ year (DateAppointed)✓ AS [TotalYears]✓ FROM tblRangers	6	
	SELECT RangerID, Name, Surname, ✓ year(Now())✓ – ✓ year (DateAppointed)✓ AS TotalYears✓ FROM tblRangers		
	SELECT RangerID, Name, Surname, ✓ 2010✓ – ✓ year (DateAppointed)✓ AS [TotalYears]✓ FROM tblRangers		
	SELECT RangerID, Name, Surname, ✓ year(DATE())✓ – ✓ year (DateAppointed)✓ AS [TotalYears]✓ FROM tblRangers		
1.4	SELECT Animal✓, format(Avg(NumAnimals) ✓, '0.00'✓) AS [AvgSighted]✓ FROM tblSightings✓ GROUP BY Animal✓	6	
	SELECT Animal✓, round(Avg(NumAnimals) ✓,2✓) AS AvgSighted✓ FROM tblSightings✓ GROUP BY Animal✓		
1.5	Input statement✓ "DELETE✓FROM tblSightings✓ WHERE SightingID = "+id✓ // **or** "+id+""	4	
1.6	UPDATE✓ tblSightings✓ SET✓ Animal = 'White Rhino'✓ WHERE Animal = 'Rhino'✓	5	
	UPDATE✓ tblSightings✓ SET✓ Animal = 'White Rhino'✓ WHERE Animal Like '%Rhino%'✓		
1.7	SELECT SightingDate,Name,Surname✓ FROM tblSightings, tblRangers✓ WHERE tblSightings.RangerID =✓ tblRangers.RangerID✓ AND Animal = 'Elephant'✓ AND SightingDate > #30/04/2010#	6	
	SELECT SightingDate,Name,Surname✓ FROM tblSightings,tblRangers✓ WHERE tblSightings.RangerID=✓ tblRangers.RangerID✓ AND Animal = 'Elephant'✓ AND month(SightingDate) > 4		
	SELECT SightingDate,Name,Surname✓ FROM tblSightings,tblRangers✓ WHERE tblSightings.RangerID=✓ tblRangers.RangerID✓ AND Animal like 'Elephant'✓ AND SightingDate > #30/04/2010#		
	TOTAL:	35	

ADDENDUM E

QUESTION 2 - JAVA: OBJECT-ORIENTED PROGRAMMING

CENTRE NUMBER:……	EXAMINATION NUMBER: ……

QUESTION 2 JAVA – MARKING GRID

QUESTION	ASPECT	MAX. MARKS	LEARNER'S MARKS
2.1			
2.1.1	**Define attributes for Competitor**: four private (1) fields, three integers named correctly (1) and one string (1)	**3**	
2.1.2	**Constructor:** name parameter (1) initialise all instance fields (1) uncomment statements give in the 'spot'-methods (1)	**3**	
2.1.3	**calculatePoints:** (3) multiply counters by correct values and add results to get total points	**3**	
2.1.4	**totalAnimals:** returns integer (1) containing sum of counters (1)	**2**	
2.1.5	**getName method:** Correct name and return type (1), Correct field returned (1)	**2**	
2.1.6	**mostSpotted method:** returns String (1) two if .. with else-statement each to determine highest count (3)	**4**	
2.1.7	**toString:** name and new line (1) and appends string (1) with category-points (1) add last(1) line with tab (1)	**5**	
2.2			
2.2.1	Declare a single Competitor object (1) Create a new File object (1) If file does not exist (1) display message and exit (1) if file exists then open file for reading (2) Read first line outside of loop (1) into name variable (1) Call constructor method of Competitor (1) using name as a parameter (1) and assign to object variable(1) initialise counters to zero (1) loop with braces correctly placed (1) Inside loop: Read line from file (1), get animal name (1) and category-letter (1) from string. Use letter and compare result (1) call appropriate method to increase count based on result (3) to handle all valid categories. Else increase invalid counter and display animal name in invalid category (1). Display number of valid and invalid entries (1)	**20**	
2.2.2	Call toString method of Competitor object (1) to display information (1)	**2**	
2.2.3	Use name of competitor to construct filename (1) ready the file for writing to new file (1) Call methods to construct output (1) and write to file (1), close the file (1)	**5**	
	TOTAL:	**49**	

ADDENDUM F

QUESTION 3: JAVA PROGRAMMING

CENTRE NUMBER: ……..	EXAMINATION NUMBER: …….		
QUESTION 3 JAVA – MARKING GRID			
QUESTION	**ASPECT**	**MAX. MARKS**	**LEARNER'S MARKS**
3.1	(2) Declare arrEntries array (1) Remove comment-signs from statements assigning strings to array	**3**	
3.2	**Option A** (1) Assign 10 to waterX and waterY (2) Heading and subheadings (1) Loop through given array Inside Loop: (2) Extract xPos and (2) yPos (2) Extract time (3) Calculate distance, (1) round down, (3) Display info in loop	**17**	
3.3	**Option B** (1) Enter the number of different types of animals (1) Loop from 1 to the number of different types (1) Inside Loop: (1) Enter animal type inside loop (1) Enter the number in the group inside loop (1) Extract first 2 letters and assign to tag (2) Extract the last letter (2) Generate number in correct range (1) Validate even number (1) Add random number and hyphen to tag (1) Heading (1) Inner loop to add the unique number of animals in each group (2) Display the correct data aligned correctly inside inner loop	**16**	
	TOTAL:	**36**	