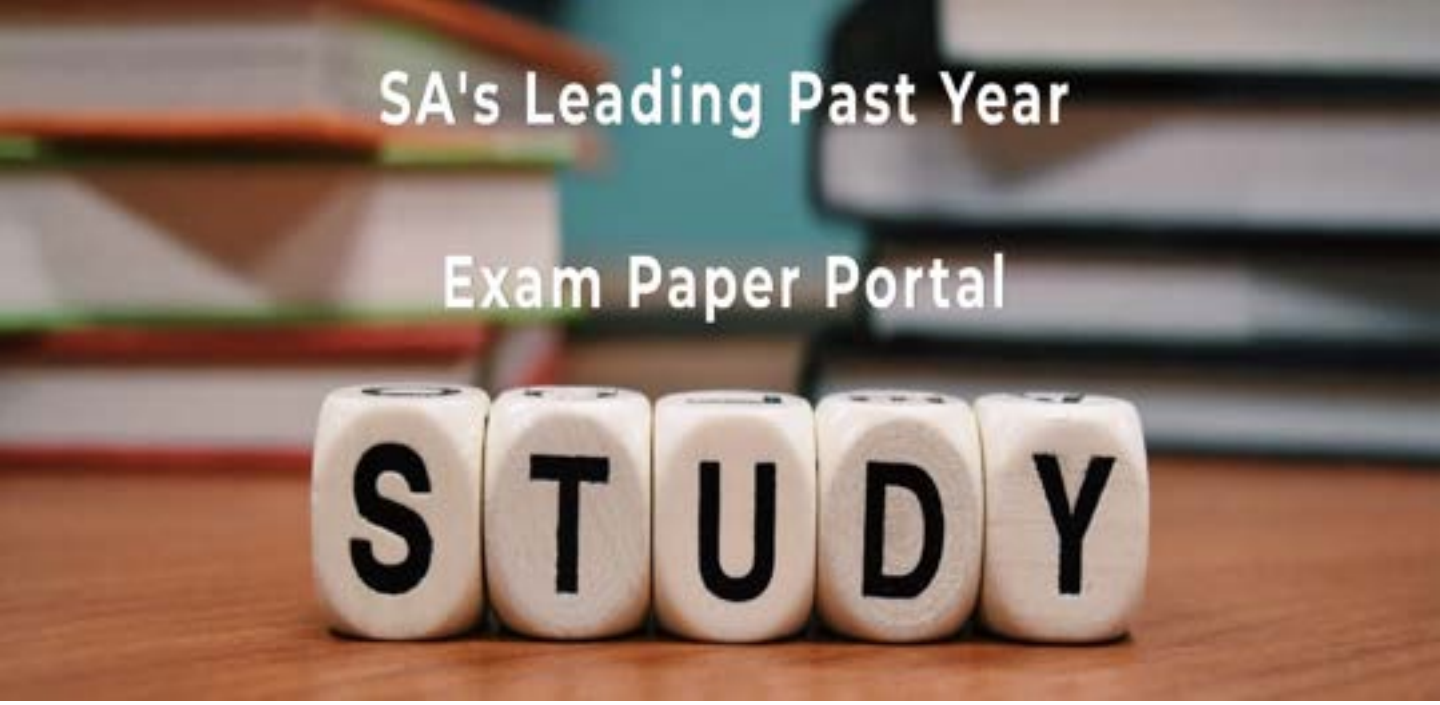

You have Downloaded, yet Another Great Resource to assist you with your Studies ☺

Thank You for Supporting SA Exam Papers

Your Leading Past Year Exam Paper Resource Portal

Visit us @ www.saexampapers.co.za

SA EXAM PAPERS

basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

NOVEMBER 2011

POSSIBLE ANSWERS

MARKS: 120

This memorandum consists of 27 pages.

Copyright reserved

GENERAL INFORMATION:

- **Pages 2–11 contain the Delphi memoranda of possible solutions for QUESTIONS 1 to 3 in programming code.**
- **Pages 12–22 contain the Java memoranda of possible solutions for QUESTIONS 1 to 3 in programming code.**
- **Pages 23–30 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
  TfrmRec = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    btnA: TButton;
    btnB: TButton;
    btnC: TButton;
    btnD: TButton;
    btnE: TButton;
    btnF: TButton;
    btnG: TButton;
    BitBtn1: TBitBtn;
    qryRec: TADOQuery;
    tblRecAg: TDataSource;
    grdRec: TDBGrid;

    procedure btnAClick(Sender: TObject);
    procedure btnBClick(Sender: TObject);
    procedure btnCClick(Sender: TObject);
    procedure btnDClick(Sender: TObject);
    procedure btnEClick(Sender: TObject);
    procedure btnFClick(Sender: TObject);
    procedure btnGClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmRec: TfrmRec;
implementation

{$R *.dfm}
```

See ADDENDUM A for alternatives and marking guidelines

```
procedure TfrmRec.btnAClick(Sender: TObject);
begin
  qryRec.Active := False;                                    //QUESTION 1.1
  qryRec.SQL.Text := 'SELECT *✓ FROM tblDams✓ ORDER BY HeightOfWall✓ ASC';
  qryRec.Active := True;
end;                                                                    (3)
//==============================================================================
procedure TfrmRec.btnBClick(Sender: TObject);                 //QUESTION 1.2
var
  pr : String;
begin
  qryRec.Active := False;
  pr := InputBox('Large Towns', 'Enter the name of the province', ''); ✓
  qryRec.SQL.Text := 'SELECT TownName, Population✓ FROM tblTowns WHERE✓
       Population > 100000✓ AND✓ Province = "' + pr + '"'✓;
    qryRec.Active := True;
end;                                                                    (6)
//==============================================================================
procedure TfrmRec.btnCClick(Sender: TObject);                 //QUESTION 1.3
begin
  qryRec.Active := False;
  qryRec.SQL.Text := 'SELECT DamID, DamName✓, (YEAR(NOW())✓ - YearCompleted✓)
       AS Age✓, ROUND (DamLevel / Capacity * 100✓, 1✓) AS Percentage✓ FROM
       tblDams';
  qryRec.Active := True;
end;                                                                    (7)
//==============================================================================
procedure TfrmRec.btnDClick(Sender: TObject);                 //QUESTION 1.4
begin
  qryRec.Active := False;
  qryRec.SQL.Text := 'SELECT Province✓, COUNT(*)✓ AS CriticalTowns✓ FROM
       tblTowns WHERE WaterRestrictions = TRUE✓ GROUP BY Province✓';
  qryRec.Active := True;
end;                                                                    (5)
//==============================================================================
procedure TfrmRec.btnEClick(Sender: TObject);                 //QUESTION 1.5
begin
  qryRec.Active := False;
  qryRec.SQL.Text := 'SELECT DISTINCT Province✓ FROM tblTowns✓, tblDams✓ WHERE
     tblTowns.DamID✓ = tblDams.DamID✓ AND River✓ = "Vaal River"✓';
  qryRec.Active := True;
end;                                                                    (7)
//==============================================================================
procedure TfrmRec.btnFClick(Sender: TObject);                 //QUESTION 1.6
begin
  qryRec.Active := False;
  qryRec.SQL.Text := 'UPDATE tblTowns✓ SET✓ WaterRestrictions = True✓ WHERE
      Province = "North West"✓';
  qryRec.ExecSQL;
  MessageDlg('Records Processed Successfully',mtInformation,[mbok],0);
end;                                                                    (4)
//==============================================================================
procedure TfrmRec.btnGClick(Sender: TObject);                 //QUESTION 1.7
begin
  qryRec.Active := False;
  qryRec.SQL.Text := 'DELETE✓ FROM tblDams✓ WHERE HeightOfWall < 11.50✓';
  MessageDlg('Records Processed Successfully',mtInformation,[mbok],0);
  qryRec.ExecSQL;
end;                                                                    (3)
//==============================================================================
end.                                                                   [35]
```

QUESTION 2: OBJECT-ORIENTED PROGRAMMING

```
unit uHouseholdXXXX;

interface
uses SysUtils;
   type
       arrType = array[1..7] of integer;
       THousehold = class (TObject)
         private
            fAccount        :string;
            fMembers        :integer;
            fArrWaterUse    :arrType;
         public
          constructor create(aAccount : string; aMembers :integer;arrWaterUse :
                                                          arrType );
          function calculateTotal:integer;
          function calculateAve:double;
          function determineHighDay:integer;
          function determineHighRisk(dayLimit:real):boolean;
          function toString:string;
       end;

implementation
//=============================================================================
// Q 2.1.1                                  (3)
constructor THousehold.create(aAccount : string; aMembers:integer;
                                            arrWaterUse:arrType);
begin
  fAccount  := aAccount; ✓
  fMembers  := aMembers; ✓
  fArrWaterUse := arrWaterUse; ✓
end;
```

Q 2.1.1
(3) Assign parameters to private fields

```
Accept a loop to assign the array
Subtract only 1 mark if the assignment statements are reversed, e.g.
aAccount := fAccount
```

```
//=====================================================================
// Q 2.1.2                                  (4)
Ignore any errors in definition (declaration) of method - no marks
Total (or return value) can be double or int

function THousehold.calculateTotal:integer;
var
  iTotal, k :integer;

begin
  iTotal := 0; ✓
  for k := 1 to length(fArrWaterUse) do ✓
     iTotal := iTotal + fArrWaterUse[k];
      // or inc(iTotal, fArrWaterUse[k]);
  result := iTotal; ✓
end;
```

Q 2.1.2
(1) Initialise total
(1) for loop
(1) Add array element to total
(1) return total (use result or function name)

```
Accept: iTotal as an instance/global variable.
Accept: loop to <=7 or < 8
Accept: adding individual elements - no loop
Accept: not using a variable iTotal - add up and assign to result- all in one
statement
```

Award 4 marks if method/code done correctly but in the main unit

```
//==========================================================================
// Q 2.1.3                                    (2)

function THousehold.calculateAve:double; ✓
begin
        result := calculateTotal / 7; ✓
end;
```

Q 2.1.3
(1) Data type of return value real (or double)
(1) Correct calculation

Accept the use of iTotal only if calculateTotal has been called (can be called in main unit.
Accept if values are added here to get a total.
Accept integer as a return type.

Award 2 marks if method/code done correctly but in the main unit

```
//==========================================================================
// Q 2.1.4                                    (8/2 = 4) (rounded up)

function THousehold.determineHighDay:integer; ✓
var
  iHighDay,  iHighAmount, k  :integer;
begin
  iHighDay := 1; ✓
  iHighAmount := fArrWaterUse[1]; ✓
  for k := 2 to 7 do✓
  begin
    if (fArrWaterUse [k] > iHighAmount) ✓ then
     begin
       iHighDay := k; ✓
       iHighAmount := fArrWaterUse[k]; ✓
     end;
     result := iHighDay; ✓
  end;
 end;
```

Q 2.1.4
(1) Return type integer
(1) Initialise iHighDay
(1) Initialise iHighAmount
(1) For loop
(1) if statement
(1) change iHighDay
(1) change iHighAmount
(1) return iHighDay

Accept sorting the amounts, also returned the correct day (full marks)
Accept correct variations of finding highest e.g. start with 0 as highest instead of first element.
Sorting done correctly but correct day not found and returned - 3 out of 4 marks

Award 4 marks if method done correctly but in the main unit

```
//==========================================================================
// Q 2.1.5                                    (9)
function  THousehold.determineHighRisk(dayLimit:real):boolean;
var
  rAve          :real;
  iCount, k     :integer;
begin
       rAve := calculateAve;
       iCount := 0; ✓
       for k := 1 to length(fArrWaterUse) do✓
       begin
         if(fArrWaterUse[k] > dayLimit) then✓
                 inc(iCount); ✓
       end;
       if ((rAve > dayLimit) ✓  OR✓ (iCount > 2)) ✓ then
        result := true✓
```

Q 2.1.5
(1) Initialise iCount
(1) Loop
(1) if array element > dayLimit
(1) increment count
(3) if rAve > dayLimit or iCount > 2
(1) return true
(1) else return false

```
      else
       result := false; ✓
end;
```

```
Accept variables as global
Do not deduct a mark for input of dayLimit
Accept: if ((calculateAve > dayLimit) OR (iCount > 2))
Accept: a single statement that returns a Boolean value
Result = ✓ (rAve > dayLimit)✓ OR ✓(iCount > 2) ✓✓
Accept: Initialising a Boolean variable, return the Boolean variable
```

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

// Q 2.1.6 (6)

```
1 mark for each piece of information = 5 marks
1 mark for adding all the information in one string
```

```
function THousehold.toString:string;
var
   sObjStr:  string;
   k:integer;
begin
   sObjStr := 'Account number : ' + fAccount + #13 + 'Number of members : ' +
           IntToStr(fMembers) + #13;
   sObjStr := sObjStr + 'Daily water usage' + #13 ✓+ 'Days:          ' + #9;
    for k := 1 to 7 do
            sObjStr := sObjStr + intToStr(k) ✓ + #9;

    sObjStr := sObjStr + #13 + 'Water used:'✓ + #9;
    for k := 1 to length(fArrWaterUse) do✓
     sObjStr := sObjStr + IntToStr(fArrWaterUse[k])✓+
                                                    #9;
                              // Join strings✓
    result :=  sObjStr;
end;
```

Q 2.1.6
(1) Headings + new line (#13 or #10)
(1) Day numbers
(1) Heading
(2) Values from array
(1) Strings concatenated

```
Accept separate array entries instead of the loop.
Accept any correct form of joining all correct information
```

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

unit Question2XXXX_U;

```
interface

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

type
  TfrmHousehold = class(TForm)
    MainMenu1: TMainMenu;
    OptionA: TMenuItem;
    OptionB: TMenuItem;
    redOutput: TRichEdit;
    OptionC: TMenuItem;
    Quit: TMenuItem;
    procedure FormActivate(Sender: TObject);
    procedure QuitClick(Sender: TObject);
    procedure OptionAClick(Sender: TObject);
    procedure OptionBClick(Sender: TObject);
    procedure OptionCClick(Sender: TObject);

  private
```

```
  public
    { Public declarations }
  end;

var
  frmHousehold: TfrmHousehold;

implementation
uses
  uHouseholdXXXX;
/==========================================================================
```

// Q 2.2.1 (2)

```
var
  Household    :THousehold; ✓
  sAccount     :string;
  iMembers     :integer;
  arrWaterUse :arrType = (481, 438, 454, 353, 421, 396, 432);
{$R *.dfm}

procedure TfrmHousehold.FormActivate(Sender: TObject);
begin
     sAccount := 'AC-23245';
     iMembers := 4;
     Household := THousehold.create(sAccount, iMembers, arrWaterUse); ✓
end;
```

Q 2.2.1
(2) Declare object variable

Deduct 1 mark for no parameters.

```
procedure TfrmHousehold.QuitClick(Sender: TObject);
begin
  Application.Terminate;
end;
//==========================================================================
```

// Q 2.2.2 (4)

```
procedure TfrmHousehold.OptionAClick(Sender: TObject);
begin
   redOutput.Clear;
   redOutput.Lines.Add(Household.toString); ✓
   redOutput.Lines.Add('');
  redOutput.Lines.Add('Total water usage: '✓ +
          IntToStr(Household.calculateTotal) ✓+' litres');
   redOutput.Lines.Add('Average water usage per day: ' +
        FloatToStrF(Household.calculateAve, ✓ ffFixed,8,1) + ' litres');
end;
```

Q 2.2.2
(1) Call the toString method of the object
(1) Display label
(1) Call calculateTotal method
(1) Call calculateAverage method

Do not be strict in the wording of the labels and formatting of values

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

// Q 2.2.3 (6)

```
procedure TfrmQuestion2.mnuOptionBClick(Sender: TObject);
var
  rAve :real;
  k :integer;
begin
     redOutput.Clear;
     rAve := Household.calculateAve; ✓
     redOutput.Lines.Add('Days and amount of water exceeding the average');
     redOutput.Lines.Add('========================================');
```

Q 2.2.3
(1) Call calculateAve method
(1) Display average
(1) Loop
(1) if
(2) Display number & difference

```
    redOutput.Lines.Add('Average water usage per day: ' +
        FloatToStrF(Household.calculateAve, ffFixed, 8, 1) ✓ + ' litres');
    redOutput.Lines.Add('Days    Value exceeding average by (litres)');
    for k := 1 to length(arrWaterUse) do✓
      begin
        if (arrWaterUse[k] > rAve) then✓
          begin
            redOutput.Lines.Add(IntToStr(k) ✓ + #9 +
            FloatToStrF(arrWaterUse[k]- rAve, ✓ ffFixed,8,1));
          end;
      end;
end;
```

```
No marks for headings
Display average - no matter how average is obtained, mark is not for
formatting
Fourth mark goes for calculation, not formatting
```

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

// Q 2.2.4 **(5)**

Q 2.2.4
(1) Input rDayLimit
(1) Call toString
(1) Call calculateHighDay
(1) If statement
(1) Display correct message

```
procedure TfrmQuestion2.mnuQuitClick(Sender: TObject);
var
  rDayLimit :double;
begin
  redOutput.Clear;
  rDayLimit := StrToFloat(InputBox('Water Limit',
                        'Enter the limit of water per day', '')); ✓
  redOutput.Lines.Add(Household.toString); ✓
  redOutput.Lines.Add('');
  redOutput.Lines.Add('The day on which the most water was used is: ' +
                  intToStr(household.determineHighDay)); ✓
  redOutput.Lines.Add('');
  if (Household.determineHighRisk(rDayLimit)) ✓ then
      redOutput.Lines.Add('High-risk household')
  else
      redOutput.Lines.Add('Not a high-risk household'); ✓
end;
end.
```

```
rDayLimit - integer or real
Second mark: For call of toString - no other way accepted to display
Third mark goes for calling method, not label. Accept with no label
Fourth mark: for calling the method as part of an if or assign to variable
Fifth mark: displaying message - mark for two messages with else or second
if
```

[45]

QUESTION 3: DELPHI PROGRAMMING

NOTE: This is only a sample – learners may answer this question in any way they see fit. Make use of the generalised 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 btnAClick(Sender: TObject);
    procedure btnBClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  frmQuestion3: TfrmQuestion3;
  iCountRefs  : integer;
  arrRefs, arrQueries : array[1..100] of String;
implementation
{$R *.dfm}
//==============================================================================
```

//QUESTION 3.1

```
procedure CreateSuggestionsFile;
var
  TFile : textfile;
begin
  AssignFile(TFile, 'Suggestions.txt');
  ReWrite(TFile);
  CloseFile(TFile);
end;
//==============================================================================
```

Q 3.1
Code was given in Afrikaans Java version.
2 marks allocated in Question 3.3

//QUESTION 3.2 (6)

```
function  validateAccNum(sAccNum:String): boolean; ✓
var
  bValid : boolean;
begin
      bValid := false; ✓
      if (length(sAccNum) = 7) ✓ and (sAccNum[1]in ['A'..'Z']) ✓ then
        begin
            bValid := true; ✓
        end;
      result := bValid; ✓
end;
```

Q 3.2
(1) Subprogram heading
(1) Initialise Boolean value
(2) if statement
(1) Change Boolean value
(1) Return Boolean value

```
Accept: if … else instead of initializing Boolean
Accept: Any correct code to obtain the first character
Accept: One statement in method returning Boolean, e.g.
Result := (length(sAccNum)and ……..);
```

```
//==============================================================================
//QUESTION 3.3                    (24) + 2
procedure TfrmQuestion3.btnAClick(Sender: TObject);
var
  inFile, sugFile : textfile;
  sLine, sAccNum, sQuery, sDate, sQueryType, sRefNum : String;
  iLoop, iComCount, iAccCount : integer;
begin
  CreateSuggestionsFile; ✓
  AssignFile(inFile, 'Data.txt'); ✓
  Reset(inFile); ✓
  AssignFile(sugFile, 'Suggestions.txt');
  Append(sugFile); ✓

  iCountRefs := 0;
  iComCount := 0;
  iAccCount := 0;
  while NOT EOF (inFile) do✓
    begin
      ReadLn(inFile, sLine); ✓

      sQueryType := Copy(sLine, 1, Pos(':', sLine) - 1); ✓
      Delete(sLine, 1, Pos(':', sLine));

      sAccNum := Copy(sLine, 1, Pos(':', sLine) - 1); ✓
      Delete(sLine, 1, Pos(':', sLine));

      sDate := Copy(sLine, 1, Pos('#', sLine) - 1); ✓
      Delete(sLine, 1, Pos('#', sLine));

      sQuery := sLine; ✓
      if (validateAccNum(sAccNum)) then✓
        begin
          if (sQueryType = 'Suggestion') then✓
             WriteLn(sugFile, sAccNum + ':' + sDate + '#' + sQuery); ✓
          else
            begin
              Inc(iCountRefs); ✓
              sRefNum := sQueryType[1]; ✓
              if (sQueryType = 'Complaint') then✓
                begin
                  Inc(iComCount); ✓
                  sRefNum := sRefNum + IntToStr(iComCount); ✓
                end
              else if (sQueryType = 'Account') then
                begin
                  Inc(iAccCount); ✓
                  sRefNum := sRefNum + IntToStr(iAccCount); ✓
                end;

              sRefNum := sRefNum + '-' + sAccNum + '-' + sDate; ✓
              arrRefs[iCountRefs] := sRefNum; ✓
              arrQueries[iCountRefs] := sQuery; ✓
            end;
        end;
       btnB.Enabled := true;
    end;

  redOutput.Lines.Clear;
  redOutput.Lines.Add('Reference Numbers');
  redOutput.Lines.Add('================');
```

Q 3.3
(1) Call createSuggestions file
(1) Open file for writing
(2) Open file Data.txt
(1) While not eof
(1) Read a line
(1) Extract type of issue
(1) Extract account Num
(1) Extract date
(1) Extract issue
(1) Call validateAccNo
(1) Check if suggestion
(1) Write suggestion to file
(1) Inside else Increase ref number counter
(1) Extract first letter of issue
(1) Check category
(2) Create ref number for complaint
(2) Create ref number for Account query
(1) Create issue reference number
(1) Store reference number in array
(1) Store query in array
(2) Display ref numbers
(1) Close Suggestions file

```
  for iLoop := 1 to iCountRefs do✓
    begin
      redOutput.Lines.Add(arrRefs[iLoop]); ✓
    end;
  CloseFile(sugFile); ✓
  CloseFile(inFile);
end;
```

Accept: Open and close Suggestion file inside loop.
While reading from file with begin and end = 1 mark, no marks with no begin and end
Accept any part of the text written to the Suggestions file.
Accept the whole word for checking purposes.

//==========

//QUESTION 3.4 (8)

```
procedure TfrmQuestion3.btnBClick(Sender: TObject);
var
  sAccNum : String;
  iLoop   : integer;
  bFound  : boolean;
begin
  sAccNum := Uppercase(InputBox('Search Queries',
            'Enter the account number', ''));
  redOutput.Lines.Clear;
  bFound := false; ✓
  if NOT(validateAccNum(sAccNum)) then✓
      ShowMessage('Invalid account number') ✓
  else
    begin
      for iLoop := 1 to iCountRefs do✓
        begin
          if (Pos(sAccNum, arrRefs[iLoop]) > 0) ✓  then
            begin
              redOutput.Lines.Add(arrRefs[iLoop] + #9 +
                                  arrQueries[iLoop]);✓
              bFound := true; ✓
            end;
        end;
      if bFound = false then
        begin
         redOutput.Lines.Add('No issues have been reported for
                  account number: ' + sAccNum);  ✓
        end;
    end; // else
 end;
```

Q 3.4
(1) Initialise Boolean variable
(1) Validate acc number
(1) Display message if invalid acc num is entered
(1) Loop
(1) Check if num entered in array
(1) Display ref num and query
(1) Change Boolean value
(1) Display message if input value not found

Do not subtract mark if no uppercase
Accept: Extract the account number and then compare

```
end.
```

//==========

[40]

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 TestDams
{
    public static void main (String[] args) throws SQLException,IOException
   {
BufferedReader inKb = new BufferedReader (new InputStreamReader
(System.in));
    Dams DB = new Dams();
    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 tblDams✓ ORDER BY HeightOfWall✓ ASC";
                  DB.query(sql);
                  break;
               }                                                          (3)
//==============================================================================
             case 'B':                                        //QUESTION 1.2
               {
                  System.out.print("Enter the name of the province : ");
                  String pr = inKb.readLine();✓
                  sql = "SELECT TownName, Population✓ FROM tblTowns WHERE✓
                      Population > 100000✓ AND✓ Province = '" + pr + "'"✓;
                  DB.query(sql);
                  break;
               }                                                          (6)
//==============================================================================
             case 'C':                                        //QUESTION 1.3
               {
                  sql = "SELECT DamID, DamName✓, (YEAR(NOW())✓ -
                  YearCompleted✓) AS Age✓, ROUND (DamLevel / Capacity *
                  100✓, 1✓) AS Percentage✓ FROM tblDams ";
```

See ADDENDUM D for alternatives and marking guidelines

```
                    DB.query(sql);
                    break;
                  }                                                    (7)
//==========================================================================
                case 'D':                                   //QUESTION 1.4
                  {
                    sql = "SELECT Province✓, COUNT(*)✓ AS CriticalTowns✓ FROM
                    tblTowns WHERE WaterRestrictions = TRUE✓ GROUP BY
                                    Province✓";
                    DB.query(sql);
                    break;
                  }                                                    (5)
//==========================================================================
                case 'E':                                   //QUESTION 1.5
                  {
                    sql = "SELECT DISTINCT Province✓ FROM tblTowns✓, tblDams✓
                     WHERE tblTowns.DamID✓ = tblDams.DamID✓ AND River✓
                        = 'Vaal River'✓";
                    DB.query(sql);
                    break;
                  }                                                    (7)
//==========================================================================
                case 'F':   //QUESTION 1.6
                  {
                    sql = "UPDATE tblTowns✓ SET✓ WaterRestrictions = True✓
                           WHERE Province = 'North West'✓";
                    DB.query(sql);
                    break;
                  }                                                    (4)
//==========================================================================
                case 'G':                                   //QUESTION 1.7
                  {
                    sql = " DELETE✓ FROM tblDams✓ WHERE HeightOfWall < 11.50✓";
                    DB.query(sql);
                    break;
                  }                                                    (3)
//==========================================================================
            }
        }while (choice != 'Q');
        DB.disconnect();
        System.out.println("Done");
      }
    }
//==========================================================================
```

[35]

QUESTION 2: OBJECT-ORIENTED PROGRAMMING

HouseholdXXXX.java

```
public class HouseholdXXXX
{
    private String account;
    private int members;
    private int [] arrWaterUse;

    public HouseholdXXXX()
    {

    }
```

// Q 2.1.1 (3)

```
    public HouseholdXXXX(String Account, int Members, int [] arrWater)
    {
        account = Account; ✓
        members = Members; ✓
        arrWaterUse = arrWater; ✓
    }
```

Q 2.1.1
(3) Assign parameter values to private fields

```
Accept a loop to assign the arrays
Subtract only 1 mark if the assignment statements are reversed, e.g.
arrWater := arrWaterUse
```

//==========

// Q 2.1.2 (4)

```
Ignore any errors in definition (declaration) of method - no marks
Return type can be double or int
```

```
public int calculateTotal()
    {
        int total = 0; ✓
        for (int k = 0 ; k < arrWaterUse.length; k++)✓
        {
            total = total + arrWaterUse[k]; ✓
            // or total += arrWaterUse[k];
        }
        return total; ✓
    }
```

Q 2.1.2
(1) Initialise total
(1) for loop
(1) Add array element to total
(1) return total

```
Accept: total as an instance / global variable.
Accept: loop to <=6 or < 7
Accept: adding individual elements - no loop
Accept: not using a variable total - add up and return in one statement
```

```
Award 4 marks if method/code done correctly but in the test/driver class
```

//==========

// Q 2.1.3 (2)

```
    public double✓ calculateAve()
    {
        return calculateTotal()/ 7.0; ✓

    }
```

Q 2.1.3
(1) Data type of return value double/int
(1) Correct calculation

Accept the use of total only if calculateTotal() has been called (can be called in test / driver class).
Accept if values are added here to get a total.
Accept int as a return type - accept / 7 instead of /7.0

Award 2 marks if method/code done correctly but in the test/driver class

//==========

// Q 2.1.4 (8/4=4) (rounded up)

```
public int determineHighDay()✓
  {
     int highDay = 1; ✓
     int highAmount = arrWaterUse[0]; ✓
     for (int k = 1;k < 7; k++)✓
     {
        if (arrWaterUse[k] > highAmount) ✓
        {
           highDay = k+1; ✓
           highAmount = arrWaterUse[k]; ✓
        }
     }
     return highDay; ✓
  }
```

Q 2.1.4
(1) Return type int
(1) Initialise highDay
(1) Initialise highAmount
(1) for loop
(1) if statement
(1) Change highDay
(1) Change highAmount
(1) return highDay

Accept sorting the amounts, also returned the correct day(full marks)
Accept correct variations of finding highest e.g. start with 0 as highest instead of first element.
Sorting done correctly but correct day not found and returned – 3 out of 4 mark)

Award 4 marks if method done correctly but in the test/driver class

//==========

// Q 2.1.5 (9)

```
public boolean determineHighRisk(double dayLimit)
  {
     double ave = calculateAve();

     int count = 0; ✓
     for (int k = 0; k < arrWaterUse.length;k++)✓
     {
        if(arrWaterUse[k] > dayLimit) ✓
        {
           count++;✓
        }
     }
     if (ave > dayLimit✓ || ✓count > 2) ✓

        return true; ✓
     else
        return false; ✓

  }
```

Q 2.1.5
(1) Initialise count
(1) Loop
(1) if array element > dayLimit
(1) Increment count
(3) if ave > dayLimit or counter > 2
(1) return true
(1) else return false

Accept variables as global/instance
Do not deduct a mark for input of dayLimit
Accept: if (calculateAve() > dayLimit || count > 2)
Accept: a single statement that returns a Boolean value
return ✓ (ave > dayLimit✓ || ✓count > 2) ✓✓
Accept: Initialising a Boolean variable, return the Boolean variable

//==========

// Q 2.1.6 **(6)**

1 mark for each piece of information = 5 marks
1 mark for adding all the information in one string

```
public String toString()
    {
        String objStr = "Account number: " + account + "\n";
        objStr = objStr + "Number of members: " + members + "\n";
        objStr = objStr + "Daily water usage" +"\n" ✓+ "Days:     " + "\t";
        for (int k = 1 ; k <= 7;k++)
        {
           objStr = objStr + k✓ + "\t";
        }
        objStr = objStr + "\n" + "Water used:"✓+ "\t";
        for (int k =0  ; k < arrWaterUse.length;k++)✓
        {
           objStr = objStr + (arrWaterUse[k] ✓ + "\t");
         }
                                // Add strings✓
        return objStr;
    }
```

Q 2.1.6
(1) Headings + new line
(1) Day numbers
(1) Heading
(2) Values from array
(1) Strings concatenated

Accept correct use of formatter to construct the string (Java)
Accept separate array entries instead of the loop.
Accept any correct form of joining all correct information

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

TestQuestion2XXXX

```
import java.util.Scanner;

public class TestQuestion2XXXX
{
//==========================================================================
```

// Q 2.2.1 **(2)**

```
public static void main(String args[]) throws Exception
{
  String accountNumber = "AC-23245";
  int members = 4;
  int [] arrWaterUse =  {481, 438, 454, 353, 421, 396, 432};

  HouseholdXXXX household ✓= new HouseholdXXXX (accountNumber, members,
         arrWaterUse); ✓
```

Q 2.2.1
(2) Declare object variable

Deduct 1 mark for no parameters.

```
//==========================================================================
   Scanner input = new Scanner(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("       Option C ");
      System.out.println(" ");
      System.out.println("       Q - QUIT");
```

```
      System.out.println(" ");
      System.out.print("        Your choice? ");
      ch = input.nextLine().toUpperCase().charAt(0);

      switch (ch)
      {
//==============================================================================
```

// Q 2.2.2 **(4)**

Q 2.2.2
(1) Call the toString method of the object
(1) Display label
(1) Call calculateTotal method
(1) Call calculateAverage method

```
      case 'A':
      {
        System.out.println();
        System.out.println(household.toString());✓
        System.out.println(" ");
        System.out.println("Total water usage: " ✓+
                            household.calculateTotal()✓ + " litres");
        System.out.printf("%s%6.1f%s","Average water usage:",
                             household.calculateAve()✓, " litres\n");
        break;
      }
```

Accept: Call to the toString method as: System.out.println(household) Do not be strict on the wording of labels or formatting of values

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

// Q 2.2.3 **(6)**

```
    case 'B':
    {
      System.out.println();
      double ave = household.calculateAve();✓
      System.out.println("Days and amount of water exceeding the average ");
      System.out.println("=============================================
      System.out.printf("%s%.1f%s","Average water usage per
            day:",household.calculateAve()✓," litres\n");
      System.out.println("Days    Value exceeding average by (litres)");

      for (int k = 0 ; k < arrWaterUse.length;k++)✓
       {
        if (arrWaterUse[k] > ave) ✓
        {
            System.out.printf("%d%s%.1f%s",(k+1)✓,
             "\t\t" , (arrWaterUse[k]- ave, ✓ "\n"));
        }
      }
      System.out.println(" ");
      break;
    }
```

Q 2.2.3
(1) Call calculateAve() method
(1) Display average
(1) Loop
(1) if
(2) Display number & difference

No marks for headings Display average - no matter how average is obtained, mark not for formatting Fourth mark goes for calculation, not formatting

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

// Q 2.2.4 **(5)**

```
    case 'C':
     {
       System.out.println("Enter the limit of water per day");
       double dayLimit = input.nextDouble();✓
       System.out.println(household.toString());✓
       System.out.println(" ");
       System.out.println("The day on which the most water was
       used: " + household.determineHighDay());✓

       if (household.determineHighRisk(dayLimit)) ✓
       System.out.println("High-risk household");
       else
       System.out.println("Not a high-risk household"); ✓

        break;
        }
```

Q 2.2.4
(1) Input dayLimit
(1) Call toString
(1) Call calculateHighDay()
(1) If statement
(1) Display correct message

```
dayLimit - integer or real
Second mark: For call of toString - no other way accepted to display
Third mark goes for calling method, not label. Accept with no label
Fourth mark: for calling the method as part of an if or assign statement
Fifth mark: displaying message - mark for two messages with else or second
if
```

```
       case 'Q':
        {
          System.exit(0);
        } // case
      } // switch
    } // while
  } // main
} // class

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

[45]

QUESTION 3: JAVA PROGRAMMING

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

TestQuestion3XXXX.java

Q 3.1
Code was given in Afrikaans Java version.
2 marks allocated in Question 3.3

//QUESTION 3.1

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

 public class TestCallCentre
{
 public void createSuggestionsFile()
  {
    try
    {
     PrintWriter out = new PrintWriter (new FileWriter ("Suggestions.txt"));
    }

    catch(IOException e)
    {
     System.out.println("Suggestion File Error!!!" + e.getMessage());
    }
  }
```

//==========

//QUESTION 3.2 (6)

```
public static boolean validateAccNum(String accNo) ✓
    {
       boolean validNo = false;✓
       if (accNo.length() == 7 ✓&& Character.isLetter (accNo.charAt(0))) ✓
       {
          validNo =true; ✓
       }
       return validNo; ✓
    }
```

Q 3.2
(1) Method heading
(1) Initialise Boolean value
(2) if statement
(1) Change Boolean value
(1) Return value

```
Accept: if … else instead of initializing Boolean
Accept: Any correct code to obtain the first character
Accept: One statement in method returning Boolean, e.g.
return (accNo.length()&& ……..);
```

//==========

//QUESTION 3.3 (24) + 2

```
String [] refNumbers = new String [100];
String [] query = new String [100];

int countRefNumbers =0;
int countComplaints = 0;
int countAccounts = 0;

public void referenceNumbers()
    {
      createSuggestionsFile();✓
      try
       {
```

```
Scanner sc = new Scanner (new FileReader✓ ("Data.txt"));✓
while (sc.hasNext())✓
{

 String line = sc.nextLine();✓
 int psnColon = line.indexOf(":");✓
 int lastPsnColon = line.lastIndexOf(":");✓
 String accNo = line.substring(psnColon+1,lastPsnColon);✓
 int psnHash = line.indexOf("#");
 String date = line.substring(lastPsnColon+1,psnHash); ✓
 String querie = line.substring(psnHash+1); ✓

 char type =line.charAt(0); ✓
 if (validateAccNum(accNo)) ✓
 {
   if (type == 'S') ✓
   {

    try
    {
     PrintWriter out = new PrintWriter(new FileWriter
                     ("Suggestions.txt",true)); ✓
     out.println(line.substring(psnColon+1)); ✓
     out.close();✓
    }
    catch(IOException e)
    {
    System.out.println("Suggestion Error!!!"
                        +e.getMessage());
    }
   }//if

   else
    {
     type = Character.toUpperCase(type);
     switch(type)✓
      {
     case 'C': countComplaints++;✓
               refNumbers[countRefNumbers]✓ =
               "C"+countComplaints+"-"+accNo+"-"+date;✓
                break;
     case 'A': countAccounts++;✓
               refNumbers[countRefNumbers]  =
               "A" ✓+countAccounts+"-"+accNo+"-"+date;
               break;
      }//switch
     query[countRefNumbers] = querie;✓
     countRefNumbers++;✓
     }//else
    }//if
    }//while
 }//try
     catch(FileNotFoundException e)
    {
      System.out.println("Error!!!"+e.getMessage());
    }
System.out.println("Reference Numbers\n================");
for (int i = 0; i<countRefNumbers;i++)✓
 {
    System.out.println(refNumbers[i]);✓
 }//for
}
```

Q 3.3
(1) Call Create Suggestions file
(2) Open file Data.txt
(1) While not eof
(1) Read a line
(1) Extract type of issue
(1) Extract account Num
(1) Extract date
(1) Extract issue
(1) Call validateAccNo
(1) Check if suggestion
(1) Open file for writing
(1) Write suggestion to file
(1) Close file
(1) Inside else Increase ref number counter
(1) Extract first letter of issue
(1) Check category
(2) Create ref number for complaint
(2) Create ref number for Account query
(1) Create issue reference number
(1) Store reference number in array
(1) Store query in array
(2) Display ref numbers

Accept: Open Suggestion file once above while, not inside loop.
While reading from file with { } = 1 mark, no marks with no { }
Accept any part of the text written to the Suggestions file.
Accept the whole word for checking purposes.
Accept using text files instead of arrays

```
//==========================================================================
//QUESTION 3.4                                    (8)
  public void searchAccount()
  {
     Scanner kb = new Scanner (System.in);
     System.out.println("Enter the account number to query");

     String accNumber = kb.next();
     boolean found = false; ✓
     System.out.println();
     if !(validateAccNum(accNumber)) ✓
          System.out.println("Invalid account number
                                        entered");✓

    else
     {
      for (int i = 0; i< countRefNumbers; i++)✓
      {
        if (refNumbers[i].contains(accNumber)) ✓
        {
          System.out.println(refNumbers[i]+"\t"+
                                        query[i]);✓
          found =true; ✓
        }//if
      }//for

     if (!found)
     {
        System.out.println("No issues have been reported for account
                                  number:"+accNumber);  } ✓
     }
    } // else
  }
```

Q 3.4
(1) Initialise Boolean variable
(1) Validate accNumber
(1) Loop
(1) Check if num entered matches ref num in array
(1) Display ref num and query
(1) Change Boolean to true
(1) Display message if input value not found
(1) Display message if invalid acc num is

```
Accept: if(refNumbers[i].indexOf(accNumber)>-1)
Accept: Extract the account number and then compare
```

```
//==========================================================================
public static void main (String [] args)
      {
         TestCallCentre obj = new TestCallCentre();
         Scanner input = new Scanner(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 = input.nextLine().toUpperCase().charAt(0);
            boolean optionA = false;
            if (ch == 'A')
            {
               obj. referenceNumbers();
               optionA = true;
            }
            if ( ch == 'B')
            {
           if (!(optionA)
            {
               System.out.println("\n\nFirst choose Option A ");
            }
           else
            {
               obj.searchAccount();
            }

          }
          if (ch == 'Q')
          {
              System.exit(0);
          }
         } // while
      }

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

[40]

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

In general:
Subtract only 1 mark for a common error made throughout all SQL's.
If no mark allocated in memo but a mistake was made, subtract a maximum of one mark

QUESTION	ASPECT	MAX. MARKS	LEARNER'S MARKS
1.1	SELECT *✓ FROM tblDams✓ ORDER BY HeightOfWall✓ ASC	3	
1.2	Input Province✓ SELECT TownName, Population✓ FROM tblTowns WHERE✓ Population > 100000✓ AND✓ Province = "' + pr + '"'✓ **Accept:** Province LIKE **Last mark: allow for a quoted string** **"100000" incorrect, must not be quoted** **Order of selected fields not important**	6	
1.3	SELECT DamID, DamName✓, YEAR(NOW())✓ - YearCompleted✓) AS Age✓, ROUND (DamLevel / Capacity * 100✓, 1✓) AS Percentage✓ FROM tblDams **Note:** SELECT DamID, DamName ….. FROM tblDams - **(one concept, 1 mark).** **New field names(all questions)-do not penalise if not exactly same text as suggested in question.** **Accept:** YEAR(DATE()) **or** 2011 **Accept:** format(DamLevel / Capacity * 100,'0.0') **Accept: correct use of int to round down to 1 dec** Int((DamLevel / Capacity * 100)*10)/10	7	
1.4	SELECT Province✓, COUNT(*)✓ AS CriticalTowns✓ FROM tblTowns WHERE WaterRestrictions = TRUE✓ GROUP BY Province✓ **Accept:** WaterRestrictions = YES **or** NO **Accept:** Count**(Any field from table instead of *)** **Accept:** WHERE WaterRestrictions **(without = true)** **GROUP BY has to be at the end.**	5	
1.5	sql = "SELECT DISTINCT Province✓ FROM tblTowns✓, tblDams✓ WHERE tblTowns.DamID✓ = tblDams.DamID✓ AND River✓ = "Vaal River"✓ **Accept:** GROUP BY Province **at the end of the SQL statement instead of** DISTINCT Province **Accept: Inner join to join tables:** …FROM tblDams INNER JOIN tblTowns ON tblDams.DamID = tblTowns.DamID….. **Accept:** LIKE 'Vaal%' **Note: Subtract 1 mark for syntax error e.g. leaving out the table names or the dot, etc.** **Accept use of aliasses e.g. tblTowns A, tblDams B**	7	

1.6	UPDATE tblTowns✓ SET✓ WaterRestrictions = True✓ WHERE Province = "North West"✓ **Accept:** Province LIKE **Accept:** WaterRestrictions = YES **or** NO **North West must be spelt correctly, quoted**	**4**	
1.7	DELETE✓ FROM tblDams✓ WHERE HeightOfWall < 11.50✓ **Accept:** DELETE *	**3**	
	TOTAL:	**35**	

ADDENDUM B

QUESTION 2: DELPHI – OBJECT-ORIENTED PROGRAMMING

(Mark in conjunction with the comments in the model answer on pages 4 - 8)

CENTRE NUMBER:	EXAMINATION NUMBER: .....

QUESTION 2: DELPHI – MARKING GRID

QUESTION	ASPECT	MAX. MARKS	LEARNER'S MARKS
2.1			
2.1.1	**Constructor:** (3) Assign parameter values to private fields	3	
2.1.2	**calculateTotal:** (1) Initialise total (1) for loop (1) Add array element to total (1) return total	4	
2.1.3	**calculateAve:** (1) Data type of return value real (or double) (1) Correct calculation	2	
2.1.4	**determineHighDay:** (1) Return type int (1) Initialise iHighDay (1) Initialise iHighAmount (1) For loop (1) if statement (1) change iHighDay (1) Change iHighAmount (1) return iHighDay	**8/2=4 (rounded up)**	
2.1.5	**determineHighRisk:** (1) Initialise count (1) Loop (1) if array element > dayLimit (1) increment count (3) if ave > dayLimit or count > 2 (1) return true (1) else return false	9	
2.1.6	**toString:** (1)Headings + new line (1)Day numbers (1)Heading (1)Values from array (1)Strings concatenated	6	
2.2			
2.2.1	(2) Declare a single object variable	2	
2.2.2	(1) Call the toString method of the object (1) Display label (1) Call calculateTotal method (1) Call calculateAverage method	4	
2.2.3	(1) Call calculateAve method (1) Display average (1) Loop (1) if (2) Display number & difference	6	
2.2.4	(1) Input dayLimit (1) Call toString (1) Call calculateHighDay (1) If statement (1) Display correct message	5	
	TOTAL:	45	

ADDENDUM C

QUESTION 3: DELPHI PROGRAMMING

(Mark in conjunction with the comments in the model answer on pages 9 - 13)

CENTRE NUMBER:	EXAMINATION NUMBER: .....

QUESTION 3: DELPHI – MARKING GRID

QUESTION	ASPECT	MAX. MARKS	LEARNER'S MARKS
3.1	Code was given in Afrikaans Java version 2 marks re-allocated in Question 3.3		
3.2	(1) Sub-program heading (1) Initialise Boolean value (2) if statement (1) Change Boolean value (1) Return Boolean value	6	
3.3	**Option A:** (1) Call method to create Suggestion file (2) Open Data file to read from (1) Open Suggestion file to write to (1) While not eof (1) Read a line (1) Extract type of issue (1) Extract account Num (1) Extract date (1) Extract issue (1) Call validateAccNo (1) Check if suggestion (1) Write suggestion to file (1) Inside else Increase ref number counter (1) Extract first letter of issue (1) Check category (2) Create ref number for complaint (2) Create ref number for Account query (1) Create issue reference number (1) Store reference number in array (1) Store query in array (2) Display ref numbers (1) Close Suggestion file	24 + 2	
3.4	**Option B:** (1) Initialise Boolean variable (1) Validate acc number (1) Display message if invalid acc num is entered (1) Inside for loop (1) Check if num entered in array (1) Display ref num and query (1) Change Boolean value (1) Display message if input value not found	8	
	TOTAL:	40	

ADDENDUM D

QUESTION 1: JAVA – PROGRAMMING AND DATABASE

CENTRE NUMBER:	EXAMINATION NUMBER: .....

QUESTION 1: JAVA – MARKING GRID

QUESTION	ASPECT	MAX. MARKS	LEARNER'S MARKS
In general: **Subtract only 1 mark for a common error made throughout all SQL's.** **If no mark allocated in memo but a mistake was made, subtract a maximum of one mark**			
1.1	SELECT *✓ FROM tblDams✓ ORDER BY HeightOfWall✓ ASC	**3**	
1.2	Input Province✓ SELECT TownName, Population✓ FROM tblTowns WHERE✓ Population > 100000✓ AND✓ Province = '" + pr + "'"✓ **Accept:** Province LIKE **Last mark: allow for a quoted string** **100000 must not be quoted** **Order of selected fields not important**	**6**	
1.3	SELECT DamID, DamName✓, YEAR(NOW())✓ - YearCompleted✓) AS Age✓, ROUND (DamLevel / Capacity * 100✓, 1✓) AS Percentage✓ FROM tblDams **Note:** SELECT DamID, DamName ….. FROM tblDams – **(one concept, 1 mark).** **New field names(all questions)-do not penalise if not exactly same text as suggested in question.** **Accept:** YEAR(DATE()) **or** 2011 **or** YEAR(NOW) **Accept:** FORMAT (DamLevel / Capacity * 100, '0.0') **Accept: correct use of int to round down to 1 dec** Int((DamLevel / Capacity * 100)*10)/10	**7**	
1.4	SELECT Province✓, COUNT(*)✓ AS CriticalTowns✓ FROM tblTowns WHERE WaterRestrictions = TRUE✓ GROUP BY Province✓ **Accept:** WaterRestrictions = YES **or** NO **Accept:** Count**(Any field from table instead of *)** **Accept:** WHERE WaterRestrictions **(without = true)** GROUP BY **has to be at the end.**	**5**	
1.5	SELECT DISTINCT Province✓ FROM tblTowns✓, tblDams✓ WHERE tblTowns.DamID✓ = tblDams.DamID✓ AND River✓ = 'Vaal River'✓ **Accept:** GROUP BY Province **at the end of the SQL statement instead of** DISTINCT Province **Accept: Inner join to join tables:** …FROM tblDams INNER JOIN tblTowns ON tblDams.DamID = tblTowns.DamID….. **Accept:** LIKE 'Vaal%' **Note: Subtract 1 mark for error e.g. leaving out the table names or the dot, etc.** **Accept use of aliases e.g. tblTowns A, tblDams B**	**7**	

1.6	UPDATE tblTowns ✓ SET✓ WaterRestrictions = True✓ WHERE Province = 'North West'✓ **Accept:** Province LIKE **Accept: WaterRestrictions = YES or** NO North West must be spelt correctly, quoted	**4**	
1.7	DELETE✓ FROM tblDams✓ WHERE HeightOfWall < 11.50✓ **Accept:** DELETE *	**3**	
	TOTAL:	**35**	

ADDENDUM E

QUESTION 2: JAVA – OBJECT-ORIENTED PROGRAMMING

(Mark in conjunction with the comments in the model answer on pages 14 - 18)

CENTRE NUMBER:	EXAMINATION NUMBER: .....

QUESTION 2: JAVA – MARKING GRID

QUESTION	ASPECT	MAX. MARKS	LEARNER'S MARKS
2.1			
2.1.1	**Constructor:** (3) Assign parameters to private fields	3	
2.1.2	**calculateTotal:** (1) Initialise total (1) for loop (1) Add array element to total (1) return total	4	
2.1.3	**calculateAve:** (1) Data type of return value is real (or double) (1) Correct calculation	2	
2.1.4	**determineHighDay:** (1) Return type int (1) Initialise highDay (1) Initialise highAmount (1) For loop (1) if statement (1) change highDay (1) change highAmount (1) return highDay	**8/2=4 (rounded up)**	
2.1.5	**determineHighRisk:** (1) Initialise count (1) Loop (1) if array element > dayLimit (1) increment count (3) if ave > dayLimit or counter > 2 (1) return true (1) else return false	9	
2.1.6	**toString:** (2)Headings + new line (1) Day numbers (2)Heading (1)Values from array (2)Strings concatenated	6	
2.2			
2.2.1	(2) Declare a single object variable	2	
2.2.2	(1) Call the toString method of the object (1) Display label (1) Call calculateTotal method (1) Call calculateAverage method	4	
2.2.3	(1) Call calculateAve method (1) Display average (1) Loop (1) if (2) Display number & difference	6	
2.2.4	(1) Input dayLimit (1) Call toString (1) Call calculateHighDay (1) If statement (1) Display correct message	5	
	TOTAL:	45	

ADDENDUM F

QUESTION 3: JAVA – PROGRAMMING

(Mark in conjunction with the comments in the model answer on pages 19 - 22)

CENTRE NUMBER:	EXAMINATION NUMBER: .....

QUESTION 3: JAVA – MARKING GRID

QUESTION	ASPECT	MAX. MARKS	LEARNER'S MARKS
3.1	Code was given in Afrikaans Java version 2 marks re-allocated in Question 3.3		
3.2	(1) Method heading (1) Initialise Boolean value (2) if statement (1) Change Boolean value (1) Return Boolean value	6	
3.3	**Option A:** (1) Call method to create Suggestions file (2) Open Data file to read from (1) While more text to read (1) Read a line (1) Extract type of issue (1) Extract account Num (1) Extract date (1) Extract issue (1) Call validateAccNo (1) Check if suggestion (1) Open file for writing (1) Write suggestion to file (1) Close file (1) Inside else Increase ref number counter (1) Extract first letter of issue (1) Check category (2) Create ref number for complaint (2) Create ref number for Account query (1) Create issue reference number (1) Store reference number in array (1) Store query in array (2) Display ref numbers	24 + 2	
3.4	**Option B:** (1) Initialise Boolean variable (1) Validate accNumber (1) Inside loop (1) Check if num entered matches ref num in array (1) Display ref num and query (1) Change Boolean to true (1) Display message if input value not found (1) Display message if invalid acc num is entered	8	
	TOTAL:	40	

Copyright reserved