These marking guidelines consist of 23 pages.
GENERAL INFORMATION:

- These marking guidelines must be used as the basis for the marking session. They were prepared for use by markers. All markers are required to attend a rigorous standardisation meeting to ensure that the guidelines are consistently interpreted and applied in the marking of candidates' work.

- Note that learners who provide an alternate correct solution to that given as example of a solution in the marking guidelines will be given full credit for the relevant solution, unless the specific instructions in the question paper were not followed or the requirements of the question were not met.

- Annexures A, B, C and D (pages 3–9) include the marking grid for each question and a table for a summary of the learner's marks.

- Annexures E, F, G and H (pages 10–23) contain examples of a programming solution for QUESTION 1 to QUESTION 4 in programming code.

- Copies of Annexures A, B, C, D and the summary of learner's marks (pages 3–9) should be made for each learner and completed during the marking session.
## ANNEXURE A

### SECTION A

### QUESTION 1: MARKING GRID – GENERAL PROGRAMMING SKILLS

<table>
<thead>
<tr>
<th>CENTRE NUMBER:</th>
<th>EXAMINATION NUMBER:</th>
<th>QUESTION</th>
<th>DESCRIPTION</th>
<th>MAX. MARKS</th>
<th>LEARNER'S MARKS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>A learner must be penalised only once if the same error is repeated.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.1</td>
<td>Panel [1.1 – Display heading] Set the panel colour to lime ✔ Set the font colour to red ✔ Set the font size to 20 pt ✔ Set the panel caption to 'Information Technology Paper 1' ✔</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.2</td>
<td>Button [1.2 – Volume] Declaration of radius and height variables ✔ Extract the height and radius from the edit ✔ Convert both to real values ✔ Calculate volume pi * sqr(rRadius) * (rHeight -1) ✔ Display message ✔ and value ✔ formatted to one decimal ✔</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.3</td>
<td>Button [1.3 – Display factors] Declaration of suitable variables for this solution ✔ Clear the rich edit output area ✔ Initialise factor counter Randomly generate number ✔ between 5 and 50 ✔ Loop ✔ from 1 to random number ✔ (Accept variations) Test if number modulus loop variable ✔ is 0 ✔ Display the value of the loop variable ✔ Increment factor counter ✔ Test if number of factors = 2 ✔ Display the random number ✔ and a message to indicate value is prime number ✔</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.4</td>
<td>Button [1.4 – Enter line and display commands] Display line of instructions ✔ and a blank line Initialise steps counter to 0 ✔ While/For loop from 1 ✔ to length of line ✔ Extract character from line at loop-index position ✔ Use CASE or multiple IF's to test 3 characters IF 'S' ✔ Test if number of steps = 10 ✔ set message to &quot;Number of forward steps more than 10&quot; ✔ break ✔ (alternative with a while loop: AND Number of steps &lt;= 10) else ✔ Increment number of steps forward by 1 ✔ Set/display message &quot;Step forward&quot; ✔ IF 'R' set/display message to &quot;Turn right&quot; ✔ IF 'L' set/display message to &quot;Turn left&quot; ✔</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TOTAL SECTION A</td>
<td></td>
<td>40</td>
</tr>
</tbody>
</table>

NOTE: sqr(rRadius); rRadius*rRadius; Power(rRadius,2)
### QUESTION 2: MARKING GRID - DATABASE PROGRAMMING

<table>
<thead>
<tr>
<th>QUESTION</th>
<th>DESCRIPTION</th>
<th>MAX. MARKS</th>
<th>LEARNER'S MARKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1.1</td>
<td>Button [2.1.1 – Alphabetical list]</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>SQL:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SELECT * FROM tblEmployees ORDER BY Surname ASC</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Concepts:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SELECT all fields ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FROM Correct table ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ORDER BY correct field ✓ (ASC not required)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.1.2</td>
<td>Button [2.1.2 – Number of children of permanent employees]</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SQL:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SELECT Surname, FirstName, Children FROM tblEmployees WHERE Children &gt; 3 AND Permanent = TRUE</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Concepts:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SELECT all the correct fields ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FROM correct table ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>WHERE Conditions: Children &gt; 3 ✓ AND ✓ Permanent = TRUE ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.1.3</td>
<td>Button [2.1.3 – Employees paid on selected date]</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SQL:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SELECT PaymentNumber, IDNumber FROM tblEmployees, tblPayments WHERE tblEmployees.EmployeeNumber = tblPayments.EmployeeNumber AND PaymentDate = #2017/01/17#</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Concepts:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SELECT correct fields ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FROM tblEmployees ✓, tblPayments ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>WHERE clause to link tables ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AND ✓ correct condition ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOTE:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PaymentDate between # #</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PaymentDate Like “2017/01/17”</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Also accept: INNERJOIN, LEFTJOIN, aliases</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.1.4</td>
<td>Button [2.1.4 – Delete payment]</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SQL:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DELETE FROM tblPayments WHERE PaymentNumber = 110</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Concepts:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DELETE ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FROM correct table ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>WHERE correct condition ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOTE:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Accept: *, all fields names, one field name</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### QUESTION 2: MARKING GRID – CONTINUE

<table>
<thead>
<tr>
<th>Button [2.1.5 – Total net salaries per month]</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SQL:</strong></td>
<td><strong>8</strong></td>
</tr>
<tr>
<td>( \text{SELECT} \ \text{Month(PaymentDate)} \ AS \text{MonthNum}, )</td>
<td></td>
</tr>
<tr>
<td>( \text{FORMAT(SUM(GrossSalary-Deductions), &quot;Currency&quot;)} )</td>
<td></td>
</tr>
<tr>
<td>( \text{AS TotalAmountPaid} )</td>
<td></td>
</tr>
<tr>
<td>( \text{FROM tblPayments} )</td>
<td></td>
</tr>
<tr>
<td>( \text{GROUP BY} \ \text{Month(PaymentDate)} )</td>
<td></td>
</tr>
<tr>
<td><strong>Concepts:</strong></td>
<td></td>
</tr>
<tr>
<td>( \text{SELECT correct field, MONTH-function ✓} )</td>
<td></td>
</tr>
<tr>
<td>( \text{AS specified fieldname ✓ (for any one of the 2 calculated fields)} )</td>
<td></td>
</tr>
<tr>
<td>( \text{SUM ✓ calculation ✓ currency format ✓} )</td>
<td></td>
</tr>
<tr>
<td>( \text{FROM correct table ✓} )</td>
<td></td>
</tr>
<tr>
<td>( \text{GROUP BY ✓ Month(PaymentDate) ✓} )</td>
<td></td>
</tr>
<tr>
<td><strong>Subtotal: SQL</strong></td>
<td><strong>25</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Button [2.2.1 – Temporary employees]</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move to first record of tblEmployees table ✓</td>
<td></td>
</tr>
<tr>
<td>Loop while not end of table ✓</td>
<td></td>
</tr>
<tr>
<td>IF Permanent = false ✓</td>
<td></td>
</tr>
<tr>
<td>Display the surname, first name, children ✓ with tabs ✓ in richedit</td>
<td></td>
</tr>
<tr>
<td>Move to next record ✓</td>
<td></td>
</tr>
<tr>
<td><strong>NOTE:</strong></td>
<td>5</td>
</tr>
<tr>
<td>Also accept: IF Permanent = ‘false’</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Button [2.2.2 – Add an employee]</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Place table in INSERT mode ✓</td>
<td></td>
</tr>
<tr>
<td>Assign correct String values to the various data fields ✓</td>
<td></td>
</tr>
<tr>
<td>Assign correct boolean value to the correct data field ✓</td>
<td></td>
</tr>
<tr>
<td>Assign correct integer value to the correct data field ✓</td>
<td></td>
</tr>
<tr>
<td>POST the updated field values ✓</td>
<td></td>
</tr>
<tr>
<td><strong>NOTE:</strong></td>
<td></td>
</tr>
<tr>
<td>Also accept: APPEND in place of INSERT</td>
<td>4</td>
</tr>
<tr>
<td>Also accept: .UPDATERECORD or any other navigation</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Button [2.2.3 – Update deductions]</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Place table in EDIT mode ✓</td>
<td></td>
</tr>
<tr>
<td>Update the deductions field ✓ with 1% of gross salary ✓</td>
<td></td>
</tr>
<tr>
<td>POST the updated field value ✓</td>
<td></td>
</tr>
<tr>
<td><strong>NOTE:</strong></td>
<td></td>
</tr>
<tr>
<td>Also accept: .UPDATERECORD or any other navigation</td>
<td>5</td>
</tr>
</tbody>
</table>

| **Subtotal: Code constructs** | **15** |

| TOTAL SECTION B | 40 |
### QUESTION 3: MARKING GRID – OBJECT-ORIENTATED PROGRAMMING

<table>
<thead>
<tr>
<th>CENTRE NUMBER:</th>
<th>EXAMINATION NUMBER:</th>
<th>MAX. MARKS</th>
<th>LEARNER’S MARKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUESTION</td>
<td>DESCRIPTION</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.1.1 Constructor:</td>
<td>Declaration/Heading ✓with three parameters ✓ two String parameters, ✓one integer ✓ Assign parameter values to attributes ✓</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>3.1.2getNumEmployees METHOD:</td>
<td>Function heading with correct data type ✓ Result statement ✓ (result := fNumEmployees)</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>3.1.3increaseNumEmployees METHOD:</td>
<td>Procedure name and parameter ✓ fNumEmployees:= ✓ fNumEmployees+ parameter value ✓</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>3.1.4compileCode METHOD:</td>
<td>Function heading with String data type ✓ Correct data type for parameter ✓ Code ← first letter of name of restaurant ✓+ last two letters ✓ of owner name ✓+ year opened ✓ Result statement ✓ NOTE: Also accept: Procedure heading with correct parameters ✓ Return variable ✓with correct data type ✓ Code ← first letter of name of restaurant ✓+ last two letters ✓ of owner name ✓+ year opened ✓</td>
<td></td>
<td>7</td>
</tr>
</tbody>
</table>

**Subtotal: Object class [17]**
## QUESTION 3: MARKING GRID – CONTINUE

<table>
<thead>
<tr>
<th>QUESTION</th>
<th>DESCRIPTION</th>
<th>MAX. MARKS</th>
<th>LEARNER’S MARKS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>3.2.1</strong></td>
<td><strong>Button [3.2.1 – Instantiate and display object]</strong></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td></td>
<td><em>Instantiate object</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Object name = <code>classname.create</code> with arguments name of restaurant, <code>year opened</code>, <code>number of employees</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>Display the object</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RichEdit component for display <code>toString</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>NOTE:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Check order and type of arguments</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Check constructor name</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>3.2.2</strong></td>
<td><strong>Button [3.2.2 – Identification code]</strong></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Call compile code method with correct object name</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Owner name as parameter</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Display the code in the edit box</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>3.2.3</strong></td>
<td><strong>Button [3.2.3 – Add employees]</strong></td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Extract the number of employees to add</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Test if the current number of employees (value from getMethod) + employees to add &lt;= max number of employees</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Call the increaseNumEmployees method with correct parameter value</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Display the updated value for the number of employees in the edit box</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Else</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Display a suitable message in the edit box</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Subtotal: Form class</strong></td>
<td>[21]</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>TOTAL SECTION C</strong></td>
<td>38</td>
<td></td>
</tr>
</tbody>
</table>
### QUESTION 4: MARKING GRID – PROBLEM-SOLVING

<table>
<thead>
<tr>
<th>Question</th>
<th>DESCRIPTION</th>
<th>MAX MARKS</th>
<th>LEARNER’S MARKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td><strong>Button [4.1 – Populate Customer array]</strong>&lt;br&gt;Read the month from combo box ✓&lt;br&gt;Initialise counter for days in month ✓&lt;br&gt;Assign ✓ and reset file ✓&lt;br&gt;Loop through file ✓&lt;br&gt;Read line ✓&lt;br&gt;Test if the line has the selected month ✓&lt;br&gt;Increment counter ✓&lt;br&gt;Find the position of # ✓ Copy number of customers ✓&lt;br&gt;(retrieving number of customers (2 marks))&lt;br&gt;Convert to integer ✓ and store number of customers in arrCustomers ✓&lt;br&gt;using the days in month variable(counter) as index ✓&lt;br&gt;Display message to indicate array were successfully populated ✓</td>
<td>14</td>
<td></td>
</tr>
</tbody>
</table>

**NOTE:**<br>The counter (initialising, incrementing) can be replaced by using character manipulation to extract the index from the line of text

<table>
<thead>
<tr>
<th>Question</th>
<th>DESCRIPTION</th>
<th>MAX MARKS</th>
<th>LEARNER’S MARKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2</td>
<td><strong>Button [4.2 – Display]</strong>&lt;br&gt;Concepts:&lt;br&gt;Extract day of week/column (1)&lt;br&gt;Filling up spaces/incomplete first week (5)&lt;br&gt;Looping through array (2)&lt;br&gt;Counting the days (1)&lt;br&gt;Constructing line to be displayed (4)&lt;br&gt;Test if line is full and display/end of week (4)&lt;br&gt;Display remaining week(1)</td>
<td>18</td>
<td></td>
</tr>
</tbody>
</table>

**TOTAL SECTION D** | 32 |
### SUMMARY OF LEARNER’S MARKS:

<table>
<thead>
<tr>
<th>CENTRE NUMBER:</th>
<th>EXAMINATION NUMBER:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SECTION A</td>
</tr>
<tr>
<td>QUESTION 1</td>
<td>40</td>
</tr>
<tr>
<td>MAX. MARKS</td>
<td></td>
</tr>
<tr>
<td>LEARNER’S MARKS</td>
<td></td>
</tr>
</tbody>
</table>
unit Question1_U;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ComCtrls, ExtCtrls, StdCtrls, Math, Buttons;
type
  TfrmQ1 = class(TForm)
    pgcQ1: TPageControl;
    tbsQuestion1_1: TTabSheet;
    tbsQuestion1_2: TTabSheet;
    tbsQuestion1_3: TTabSheet;
    pnlQ1_1: TPanel;
    btnQ1_3: TButton;
    tbsQuestion1_4: TTabSheet;
    btnQ1_4: TButton;
    redQ1_3: TRichEdit;
    pnlBtns: TPanel;
    bmbClose: TBitBtn;
    Label1: TLabel;
    redQ1_4: TRichEdit;
    Label3: TLabel;
    Label4: TLabel;
    edtHeight: TEdit;
    edtRadius: TEdit;
    btnQ1_2: TButton;
    Label5: TLabel;
    Label6: TLabel;
    procedure pnlQ1_1Click(Sender: TObject);
    procedure btnQ1_2Click(Sender: TObject);
    procedure btnQ1_3Click(Sender: TObject);
    procedure btnQ1_4Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  frmQ1: TfrmQ1;
implementation
{$R *.dfm}
// ======================================================================
// Question 1.1  (4 marks)
// ======================================================================
procedure TfrmQ1.pnlQ1_1Click(Sender: TObject);
begin
  pnlQ1_1.Color := clLime;
  pnlQ1_1.Font.Color := clRed;
  pnlQ1_1.Font.Size := 20;
  pnlQ1_1.caption := 'Information Technology Paper 1';
end;
// ======================================================================
// Question 1.2  (9 marks)
// ======================================================================
procedure TfrmQ1.btnQ1_2Click(Sender: TObject);
var
  rRadius, rHeight: real;
  rLiquidVol: real;
begin
  rRadius := StrToFloat(edtRadius.Text);
  rHeight := StrToFloat(edtHeight.Text);
rLiquidVol := pi * sqr(rRadius) * (rHeight - 1);
ShowMessage('The volume is ' + FloatToStrF(rLiquidVol, ffFixed, 5, 1));
end;
// Question 1.3  (13 marks)
// // ======================================================================
// Question 1.4  (14 marks)
// // ======================================================================
procedure TfrmQ1.btnQ1_3Click(Sender: TObject);
var
  iNumber, I, iNumFactors: integer;
begin
  redQ1_3.Clear;
iNumFactors := 0;
iNumber := Random(50 - 5 + 1) + 5;
for I := 1 to iNumber do
begin
  if iNumber mod I = 0 then
  begin
    redQ1_3.Lines.Add(IntToStr(I));
    Inc(iNumFactors);
  end;
end;
if iNumFactors = 2 then
  redQ1_3.Lines.Add(#13 + IntToStr(iNumber) + ' is a prime number');
end;
procedure TfrmQ1.btnQ1_4Click(Sender: TObject);
var
  sCommandLine, sCommand: String;
sChar: char;
i, iNumSteps: integer;
begin
  // Provided code
  sCommandLine := upperCase(InputBox('Robot instructions',
    'Enter a line of instructions', 'SSRSLSR LLSSR'));
  redQ1_4.Lines.Clear;
  redQ1_4.Lines.Add(sCommandLine);
  redQ1_4.Lines.Add('');
iNumsteps := 0;
i := 1;
while (i <= length(sCommandLine)) AND (iNumSteps <= 10) do
begin
  sChar := sCommandLine[i];
case sChar of
  'S': begin
    if iNumsteps = 10 then
    begin
      sCommand := 'Number of forward steps more than 10';
      Break;
    end;
    else
    begin
      Inc(iNumsteps);
      sCommand := 'Step forward';
    end;
  end;
  'L': sCommand := 'Turn left';
  'R': sCommand := 'Turn right';
end;
redQ1_4.Lines.Add(sCommand);
Inc(i); 
end;

// Alternative solution 
{ for i := 1 to length(sCommandLine) do 
begin 
sChar := sCommandLine[i]; 
case sChar of 
'S': begin 
    Inc(iNumSteps); 
    sCommand := 'Step forward'; 
    if iNumSteps > 10 then 
        sCommand := 'Number of forward steps exceeds 10'; 
end;
'L': sCommand := 'Turn left'; 
'R': sCommand := 'Turn right'; 
end; 
redQ1_4.Lines.Add(sCommand); 
if iNumSteps > 10 then 
    break; 
end; 
} 
end;

// -----------------------------------------------------------------------
{$REGION 'Provided code - Do not modify'}
procedure TfrmQ1.FormCreate(Sender: TObject); 
begin 
    pgcQ1.ActivePageIndex := 0; 
    CurrencyString := 'R'; 
end; 
{$ENDREGION}
end.
ANNEXURE F: SOLUTION FOR QUESTION 2

unit Question2_U;
// --- Delphi and Database programming -------------------------------
// Possible solution for Question 2.
// ---------------------------------------------------------------------
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, ExtCtrls, ConnectDB_U, DB, ADODB, Grids,
  DBGrids, ComCtrls, DateUtils, DBCtrls;
type
  TfrmDBQuestion2 = class(TForm)
    pnlBtns: TPanel;
    bmbClose: TBitBtn;
    bmbRestoreDB: TBitBtn;
    grpTblPayments: TGroupBox;
    grpTblEmployees: TGroupBox;
    dbgEmployees: TDBGrid;
    dbgPayments: TDBGrid;
    tabsQ2_2ADO: TTabSheet;
    tabsQ2_1SQL: TTabSheet;
    btnQ2_2_1: TButton;
    redQ2: TRichEdit;
    grpresuits: TGroupBox;
    dbgrdSQL: TDBGrid;
    grpOutput: TGroupBox;
    pgcTabs: TPageControl;
    pnlTables: TPanel;
    btnQ2_1_1: TButton;
    btnQ213: TButton;
    btnQ212: TButton;
    btnQ2_1_4: TButton;
    btnQ2_1_5: TButton;
    btnQ2_2_2: TButton;
    btnQ2_2_3: TButton;
  private
    procedure bmbRestoreDBClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btnQ2_1_1Click(Sender: TObject);
    procedure btnQ213Click(Sender: TObject);
    procedure btnQ212Click(Sender: TObject);
    procedure btnQ2_1_4Click(Sender: TObject);
    procedure btnQ2_1_5Click(Sender: TObject);
    procedure btnQ2_2_1Click(Sender: TObject);
    procedure btnQ2_2_2Click(Sender: TObject);
    procedure btnQ2_2_3Click(Sender: TObject);
  public
    property bmbRestoreDB: TBitBtn read bmbRestoreDB write bmbRestoreDB;
  end;
var
  frmDBQuestion2: TfrmDBQuestion2;
dbCONN: TConnection;
// Provided global variables

  tblEmployees, tblPayments : TADOTable;

implementation
{$R *.dfm}
{$R+}
{$Region 'Question 2.1 - SQL SECTION'}
// Question 2.1.1  (3 marks)
procedure TfrmDBQuestion2.btnQ2_1_1Click(Sender: TObject);
var
  sSQL1: String;
begin
  sSQL1 := 'SELECT * FROM tblEmployees ORDER BY Surname ASC';
  // Provided code - do not change
  dbCONN.runSQL(sSQL1);
end;

// Question 2.1.2  (5 marks)
procedure TfrmDBQuestion2.btnQ212Click(Sender: TObject);
var
  sSQL2: String;
begin
  sSQL2 := 'SELECT Surname, FirstName, Children ' +
           'FROM tblEmployees WHERE Children > 3 AND Permanent = TRUE';
  // Provided code - do not change
  dbCONN.runSQL(sSQL2);
end;

// Question 2.1.3  (6 marks)
procedure TfrmDBQuestion2.btnQ213Click(Sender: TObject);
var
  sSQL3: String;
begin
  sSQL3 := 'SELECT PaymentNumber, IDNumber FROM tblEmployees E, tblPayments P ' +
           'WHERE E.EmployeeNumber = P.EmployeeNumber AND PaymentDate = 
           #2017/01/17#';
  // Provided code - do not change
  dbCONN.runSQL(sSQL3);
end;

// Question 2.1.4  (3 marks)
procedure TfrmDBQuestion2.btnQ2_1_4Click(Sender: TObject);
var
  sSQL4: String;
begin
  sSQL4 := 'DELETE * FROM tblPayments WHERE PaymentNumber = 110';
  // Provided code - do not change
  dbCONN.executeSQL(sSQL4, dbgPayments);
end;

// Question 2.1.5  (8 marks)
procedure TfrmDBQuestion2.btnQ2_1_5Click(Sender: TObject);
var
  sSQL5: String;
begin
  sSQL5 := 'SELECT Month(PaymentDate) as MonthNum, '+
           'FORMAT(SUM(GrossSalary-Deductions), "Currency") AS TotalAmountPaid ' +
           ' FROM tblPayments GROUP BY Month(PaymentDate)';
  // Provided code - do not change
  dbCONN.runSQL(sSQL5);
end;
procedure TfrmDBQuestion2.btnQ2_2_1Click(Sender: TObject);
begin
    // Provided code
    redQ2.Clear;
    redQ2.Paragraph.TabCount := 2;
    redQ2.Paragraph.Tab[0] := 80;
    redQ2.Lines.Add('Temporary employees');
    redQ2.SelAttributes.Style := [fsBold, fsUnderline];
    redQ2.Lines.Add('Surname' + #9 + 'Firstname' + #9 + 'Children');

    // Add your code here
    tblEmployees.First;
    while not tblEmployees.Eof do
    begin
        if (tblEmployees['Permanent'] = False) then
        begin
            redDisplay.Lines.Add(tblEmployees['Surname'] + #9 + tblEmployees['FirstName'] + #9 + IntToStr(tblEmployees['Children']));
        end;
        tblEmployees.Next;
    end;

    // Alternative solution
    tblEmployees.First;
    while not tblEmployees.Eof do
    begin
        if (tblEmployees.FieldByName('Permanent').AsBoolean = False) then
        begin
            redDisplay.Lines.Add(tblEmployees.FieldByName('Surname').AsString + #9 + tblEmployees.FieldByName('FirstName').AsString + #9 + tblEmployees.FieldByName('Children').AsString);
        end;
        tblEmployees.Next;
    end;
end;

procedure TfrmDBQuestion2 btnQ2_2_2Click(Sender: TObject);
begin
    tblEmployees.Insert;
    tblEmployees['Surname'] := 'Zwelini';
    tblEmployees['FirstName'] := 'Lungile';
    tblEmployees['IDNumber'] := '7601050179081';
    tblEmployees['Permanent'] := True;
    tblEmployees['Children'] := 3;
    tblEmployees.Post;

    // Alternative solution
    tblEmployees.Insert;
    tblEmployees.FieldByName('Surname').AsString := 'Zwelini';
    tblEmployees.FieldByName('FirstName').AsString := 'Lungile';
    tblEmployees.FieldByName('IDNumber').AsString := '7601050179081';
    tblEmployees.FieldByName('Permanent').AsBoolean := True;
    tblEmployees.FieldByName('Children').AsInteger := 3;
    tblEmployees.Post;
end;
procedure TfrmDBQuestion2.btnQ2_2_3Click(Sender: TObject);
begin
  tblPayments.Edit;
  tblPayments['Deductions'] := tblPayments['Deductions'] + tblPayments
                              ['GrossSalary'] * 0.01;
  tblPayments.Post;

  // Alternative solution
  // tblPayments.Edit;
  // tblPayments.FieldByName('Deductions').AsFloat :=
  //     tblPayments.FieldByName('Deductions').AsFloat +
  //     (tblPayments.FieldByName('GrossSalary').AsFloat * 0.01);
  // tblPayments.Post;
end;

procedure TfrmDBQuestion2.bmbRestoreDBClick(Sender: TObject);
begin
  dbCONN.RestoreDatabase(dbgEmployees, dbgPayments, dbgrdSQL);
  redQ2.Clear;
  tblEmployees := dbCONN.tblOne;
  tblPayments := dbCONN.tblMany;
end;

procedure TfrmDBQuestion2.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  dbCONN.dbDisconnect;
end;

procedure TfrmDBQuestion2.FormCreate(Sender: TObject);
begin
  CurrencyString := 'R';
  dbCONN := TConnection.Create;
  dbCONN.dbConnect;
  tblEmployees := dbCONN.tblOne;
  tblPayments := dbCONN.tblMany;
  dbCONN.setupGrids(dbgEmployees, dbgPayments, dbgrdSQL);
  pgcTabs.ActivePageIndex := 0;
end;

{$ENDREGION}
ANNEXURE G: SOLUTION FOR QUESTION 3

OBJECT CLASS:
// Possible solution for Question 3.1
unit Restaurant_U;
interface
uses
    SysUtils, DateUtils;
type
    TRestaurant = class(TObject)
    private
        fName: String;
        fYearOpened: String;
        fNumEmployees: integer;
    public
        constructor Create(sName, sYearOpened: String; iNumEmployees: integer);
        function toString: String;
        function getNumEmployees: integer;
        function compileCode(sOwner: String): String;
        procedure increaseNumEmployees(iValue: integer);
    end;
implementation

{ TRestaurant }

//======================================================================
// Question 3.1.1  (5 marks)
//======================================================================
constructor TRestaurant.Create(sName, sYearOpened: String; iNumEmployees: integer);
begin
    fName := sName;
    fYearOpened := sYearOpened;
    fNumEmployees := iNumEmployees;
end;

//======================================================================
// Question 3.1.2  (2 marks)
//======================================================================
function TRestaurant.getNumEmployees: integer;
begin
    Result := fNumEmployees;
end;

//======================================================================
// Question 3.1.3  (3 marks)
//======================================================================
procedure TRestaurant.increaseNumEmployees(iValue: integer);
begin
    fNumEmployees := fNumEmployees + iValue;
end;
Question 3.1.4  (7 marks)

function TRestaurant.compileCode(sOwner: String): String;
Var
  sCode: String;
begin
  sCode := fName[1] + Copy(sOwner,length(sOwner)-1) + fYearOpened;
  Result := sCode;
end;

Provided code - toString

function TRestaurant.toString: String;
var
  sResult: String;
begin
  sResult := 'Restaurant name: ' + fName + #13 + 'Year opened: ' +
             fYearOpened + #13 + 'Number of employees: ' + intToStr(fNumEmployees) + #13;
  Result := sResult;
end;

end.
MAIN (APPLICATION) CLASS:

// Possible solution for Question 3.2

unit Question3_U;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls, Restaurant_U, ComCtrls, ExtCtrls, jpeg, Spin;

type
  TfrmQ3 = class(TForm)
    GroupBox1: TGroupBox;
    Label1: TLabel;
    Label3: TLabel;
    edtCompanyName: TEdit;
    btnQ3_2_1: TButton;
    GroupBox2: TGroupBox;
    btnQ3_2_2: TButton;
    Label5: TLabel;
    edtOwnerName: TEdit;
    edtIDCode: TEdit;
    edtYearOpened: TEdit;
    Label2: TLabel;
    spnNumEmployees: TSpinEdit;
    GroupBox4: TGroupBox;
    edtAdd: TEdit;
    Label6: TLabel;
    btnQ3_2_3: TButton;
    Label7: TLabel;
    edtUpdated: TEdit;
    redQ3: TRichEdit;

  end;

var
  frmQ3: TfrmQ3;

implementation

{$R *.dfm}

//======================================================================
// Question 3.2.1  (8 marks)
// ======================================================================
procedure TfrmQ3.btnQ3_2_1Click(Sender: TObject);
begin
  redQ3.Clear;//Provided code
  objRestaurant := TRestaurant.Create(edtCompanyName.Text,
    trim(edtYearOpened.Text), spnNumEmployees.value);
  redQ3.Lines.Add(objRestaurant.ToString);
end;
// Question 3.2.2   (3 marks)
//
procedure TFormQ3.btnQ3_2_2Click(Sender: TObject);
begin
  edtIdCode.Text := objRestaurant.compileCode(edtOwnerName.Text);
end;

// Question 3.2.3   (10 marks)
//
procedure TFormQ3.btnQ3_2_3Click(Sender: TObject);
begin
  iNumEmplToAdd := StrToInt(edtAddEmployees.Text);
  if objRestaurant.getNumEmployees + iNumEmplToAdd <= iMaxEmployees then
    begin
      objRestaurant.increaseNumEmployees(iNumEmplToAdd);
      edtUpdatedEmployees.Text := IntToStr(objRestaurant.getNumEmployees);
    end
  else
    begin
      edtUpdatedEmployees.Text := 'Exceeds max';
    end;
end;

procedure TFormQ3.FormShow(Sender: TObject);
begin
  btnQ3_2_1.SetFocus;
end;
end.
ANNEXURE H: SOLUTION FOR QUESTION 4

// A possible solution for Question 4
unit Question4_U;
interface
uses
    Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
    Dialogs, StdCtrls, ComCtrls, ExtCtrls, Math, Buttons;
type
    TfrmQ4 = class(TForm)
        btnQ4_1: TButton;
        cmbDays: TComboBox;
        cmbMonths: TComboBox;
        btnQ4_2: TButton;
        Panel1: TPanel;
        Label1: TLabel;
        Panel2: TPanel;
        redQ4: TRichEdit;
        Label2: TLabel;
        bmbClose: TBitBtn;
    private
        { Private declarations }
    public
        { Public declarations }
    end;

// Provided code - declarations
const
    arrDays: array [1 .. 7] of String = ('Sun', 'Mon', 'Tue', 'Wed', 'Thu',
                                         'Fri', 'Sat');
    arrTempCustomers: array [1 .. 31] of integer = (248, 81, 189, 141, 163, 163,
                                         233, 64, 145, 188, 108, 124, 120, 130, 57, 64, 131, 54, 138,
                                         71, 75, 152, 126, 170, 56, 157, 230, 82, 199, 119, 119, 136);
var
    frmQ4: TfrmQ4;
    arrCustomers: array [1 .. 31] of integer;

// User declarations
iDaysInMonth: integer = 0;
implementation
{$R *.dfm}
{$R+}

// Question 4.1 (14 marks)
// Question 4.1 (14 marks)
procedure TfrmQ4.btnQ4_1Click(Sender: TObject);
var
    tFile: TextFile;
    sLine, sMonth: String;
iPos : integer;
begin
    iDaysInMonth := 0;
sMonth := cmbMonths.Text;
AssignFile(tFile, 'Visitors.txt');
Reset(tFile);
while NOT EOF(tFile) do
begin
    Readln(tFile, sLine);
    if Pos(sMonth, sLine) > 0 then
begin
    iDaysInMonth := iDaysInMonth + 1;
end;
end;
redQ4.Text := IntToStr(iDaysInMonth);
end;
inc(iDaysInMonth, 1);
iPos := pos('#', sLine);
arrCustomers[iDaysInMonth] := StrToInt(copy(sLine, iPos+1, length(sLine)));
end;
end;
ShowMessage('Array successfully populated.');
end;
// =====================================================================
// Question 4.2  (18 marks)
// =====================================================================
procedure TfrmQ4.btnQ4_2Click(Sender: TObject);
var
  iCnt, iDate: integer;
sOutput: String;
iRow, iDayOfWeek, iCol, iWeekLoop: integer;
sLine: String;
inNumRows: integer;
begin
  // Provided code
  redQ4.Clear;
  redQ4.SelAttributes.Style := [fsBold];
  redQ4.Lines.Add('Calendar for ' + cmbMonths.Text + #13);
  sOutput := '';
  for iCnt := 1 to 7 do
    begin
      sOutput := sOutput + arrDays[iCnt] + #9;
    end;
  redQ4.SelAttributes.Style := [fsBold];
  redQ4.Lines.Add(sOutput);
  // Question 4.2 - Type your code here
  iDayOfWeek := cmbDays.ItemIndex;
iDate := 1;
  for iCol := 1 to iDayOfWeek do
    begin
      sLine := sLine + '' + #9;
    end;
  while (iDate <= iDaysInMonth) do
    begin
      if (iDate + 7) <= iDaysInMonth then
        iWeekLoop := 7 - iDayOfWeek
      else // 1
        iWeekLoop := iDaysInMonth - iDate + 1;
      for iCnt := 1 to iWeekLoop do
        begin
          sLine := sLine + IntToStr(iDate) + ' (' + IntToStr(arrCustomers[iDate]) + ' )' + #9;
          inc(iDate);
        end;
      redQ4.Lines.Add(sLine);
      sLine := ''; iDayOfWeek := 0;
    end;
  //======================================================================
// Question 4.2 - Alternative 1
iDayOfWeek := cmbDays.ItemIndex + 1;
sLine := ''; for iCol := 1 to iDayOfWeek - 1 do
  sLine := sLine + '' + '#9; for iDate := 1 to iDaysInMonth do
    // begin
    //  sLine := sLine + Copy(arrDates[iDayOfWeek],1,2) + '(' + IntToStr(arrCustomers[iDate]) + ')' + '#9; inc(iDayOfWeek);
    // if iDayOfWeek = 8 then
    //     begin
    //       redQ4.Lines.Add(sLine);
    //       sLine := ''; iDayOfWeek := 1;
    //     end;
    // end;
    redQ4.Lines.Add(sLine);
// Question 4.2 - Alternative 2
iDayOfWeek := cmbDays.ItemIndex + 1;
iDate := 1;
iNumRows := Ceil((iDaysInMonth + iDayOfWeek) / 7);
for iRow := 1 to iNumRows do
  begin
    sLine := ''; for iCol := 1 to 7 do
      begin
        if (iRow = 1) AND (iCol < iDayOfWeek) then
          begin
            sLine := sLine + '' + '#9;
          end
        else
          if iDate <= iDaysInMonth then
            begin
              sLine := sLine + IntToStr(iDate) + '(' + IntToStr(arrCustomers[iDate]) + ')' + '#9;
              iDate := iDate + 1;
            end;
      end;
    redQ4.Lines.Add(sLine);
  end;
end.
{$REGION 'PROVIDED CODE - DO NOT MODIFY!'}
procedure TfrmQ4.FormCreate(Sender: TObject);
begin
  redQ4.Paragraph.TabCount := 7;
end;
{$ENDREGION}
end.