NATIONAL SENIOR CERTIFICATE

GRADE12

INFORMATION TECHNOLOGY P1

NOVEMBER 2017

MARKING GUIDELINES

MARKS: 150

These marking guidelines consist of 26 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 and C** (pages 3–11) include the marking grid for each question and a table for a summary of the learner’s marks.

- **Annexures D, E, and F** (pages 12–26) contain examples of a programming solution for QUESTION 1 to QUESTION 3 in programming code.

- Copies of **Annexures A, B and C** (pages 3–11) and the **summary of learner’s marks** (page 12) should be made for each learner and completed during the marking session.
### QUESTION 1: MARKING GRID – GENERAL PROGRAMMING SKILLS

**General notes:**
- A learner must be penalised only once if the same error is repeated.
- Begin and End must be marked together with the structure (Loops, If statements).
  This means: If the begin and end was not coded where required in order to work correctly, the mark for the structure (loop or if) must not be allocated.

<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>
</table>
|                |                      | 1.1      | **Procedure Form**
Set caption ✓
Set font size ✓
Set background colour of panel ✓ to lime
(Or any other colour)
Numbers representing colours allowed
**No marks** for changing the properties in the Object inspector. | 3          |                |
|                |                      | 1.2.1    | **[Button] Larger number**
Extract number 1 and number 2 as numeric values ✓
Test if number 1 > number 2
  Set the result edit box to number1 ✓
Test if number 2 > number 1
  Set the result edit box to number2 ✓
Test if number 1 = number 2
  Set the result edit box to ‘Equal’ ✓

**NOTE:** Accept
  The correct use of if..else
  The correct use of Max(Num1,Num2) | 4          |                |
|                |                      | 1.2.2    | **[Button] Swap words**
Extract word 1 and word 2 from edit boxes ✓
Store word 1 in temporary storage ✓
Assign word 1 to word 2 ✓
Assign word 2 to word in temporary storage ✓
Display both words in the edit boxes ✓

**Also accept:**
  If word 2 is stored in temp with correct code
  The use of the edit box as temporary storage

**Alternative solution:**
Extract word 1 and word 2 from edit boxes (1 mark)
Assign word 1 to word 2 (2 marks)
Assign word 2 from temporary storage/edit box (1 mark)
Display both words in the edit boxes (1 mark) | 5          |                |
### 1.3.1 Combo box

Extract index of number of cakes from combo box and add 1 ✔
(Or extract number of cakes from combo box)
Correct use of code to load an img file onto img component ✔
Correct file name ✔
Correct formula to calculate cost of cakes ✔
Display cost as currency and two decimal places ✔

**Also accept:**
The use of the value of the constant in the formula
Any acceptable way of formatting output to currency, including using R and formatting the value to two decimal places

**Note:** Ensure the correct data types are used.

### 1.3.2 Button [1.3.2 – Calculate the amount of sugar]

Correct formula to calculate the sugar in grams ✔
Display the sugar in grams in the edit box ✔
Calculate number of sugar packets to be purchased ✔ rounded up ✔
Display the number of packets of sugar ✔

**Also accept:**
4 If statements
>0 and <=1000: 1
>1000 and <=2000: 2
>2000 and <= 3000: 3
>3000 and <= 4000: 4

### 1.4.1 Radiogroup [Type of user]

If the first index or third index is selected ✔
Display panel ✔
Else (if the second index is selected) ✔
Hide panel ✔

**Guideline for marking:**
*Logical constructs* to ensure the correct use of show and hide (2 marks)
The code used to display and hide (2 marks)

**Also accept:**
Case condition
0 index – display
1 index – hide
2 index – display
| 1.4.2 | **Button [1.4.2 – Validate password]** | Set counter to 0  
Extract the password from the edit box  
Test if the length is 6 or more  
Test if the first letter is a capital letter  
Loop from (1 or 2) to length of password  
  - Check if character is a special character (from list)  
  - Increase counter for special characters  
Test if all three conditions are true (nested, flag, etc.)  
  - Output ('Valid Password')  
  - Enable button  
Else  
  - Output message ('Invalid password')  
  - Clear password field  
**Alternative:** Test for special characters using case |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TOTAL SECTION A:</strong></td>
<td><strong>55</strong></td>
<td><strong>11</strong></td>
</tr>
</tbody>
</table>

| 1.4.3 | **Button [1.4.3 – Encrypt password]** | Increments the first character to the next character  
Change 'Z' to 'A'  
Replace only the first character with new character  
Display new password  
**Also accept:** The use of the case statement |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TOTAL SECTION A:</strong></td>
<td><strong>55</strong></td>
<td><strong>5</strong></td>
</tr>
</tbody>
</table>

| 1.5.1 | **Button [1.5.1 – Perfect square]** | Use an input box to enter a number.  
Convert to a number  
Test if the square root = trunc (square root)  
(or any other correct way)  
  - Display message the number is a perfect square  
Else  
  - Display message the number is not a perfect square  
**Also accept:**  
Output message without displaying the number  
Test if the square root contains a full stop (.)  
Trunc or any function to remove the decimal part. |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TOTAL SECTION A:</strong></td>
<td><strong>55</strong></td>
<td><strong>6</strong></td>
</tr>
</tbody>
</table>

| 1.5.2 | **Button [1.5.2 – Sequence of numbers]** | *Set the display sequence variable to 1 or null*  
Sum variable = 0 or 1 – depending on solution  
Set the first number to 1  
Repeat (looping - or while)  
  - Join/display number to the display sequence  
  - Add the number to sum  
  - Multiply number to the constant variable/value of 3  
Until the sum > 1000 (Correct condition)  
Display the sequence as a compiled string or display one by one inside the loop  
The display sequence can be horizontal or vertical.  
**Also accept:**  
While sum <= 1000  
While sum < 1000  
Repeat until sum >= 1000 |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TOTAL SECTION A:</strong></td>
<td><strong>55</strong></td>
<td><strong>7</strong></td>
</tr>
</tbody>
</table>
## ANNEXURE B

### SECTION B

### QUESTION 2: MARKING GRID - OBJECT-ORIENTED 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><strong>Constructor:</strong> Constructor ✓ Create</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Three string parameters ✓ and one integer parameter ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Assign parameter values to attributes ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.1.2</td>
<td><strong>increaseIssueNr Procedure:</strong></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Procedure ✓ (Not function)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Increment fIssueNr by 1 ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Do not accept:</strong> Result := fIssueNr + 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.1.3</td>
<td><strong>resetExpiryDate Procedure:</strong></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Extract the year value from system date (sDate) ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Add 1 to the year ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Extract month and day from system date and add year ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Assign new date to fExpiryDate attribute ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Also accept:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Any other way to determine the date and increment the year</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>fExpiryDate := DateToStr(StrToDate(sDate)+365)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>fExpiryDate := DateToStr(Date+365)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.1.4</td>
<td><strong>hasExpired FUNCTION:</strong></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Convert string to date format ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Comparison ✓ of the expiry date and system date ✓ (&gt; or &lt;)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Result based on the condition ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>else</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reverse result ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Also accept &lt;= or &gt;=</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.1.5</td>
<td><strong>generateSecurityCode METHOD:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>----------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Initialise security code variable to empty string ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Create string with characters 0..9 and A..F ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>sChar := '0123456789ABCDEF' (Case, Array, String)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Loop with counter from 1 to 5 ✓ (or any applicable range)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Randomly generate value in range 1 to 16 ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>and extract character ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Repeat for second character ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Join the two characters to security code ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>If counter &lt; 5 ✓ (or any correct method to remove the last colon/not include a colon as the last character)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Join colon character to security code ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Assign security code to securityCode attribute ✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>Candidate loses two marks if the first character to be generated is always a character and the second character is always a number or the other way around.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2.1.6</th>
<th><strong>toString METHOD</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Add attributes: certificate holder, expiry date security code ✓</td>
</tr>
<tr>
<td></td>
<td>Add issueNr attribute as a string ✓</td>
</tr>
<tr>
<td></td>
<td>Any code to display attributes in columns ✓ e.g. #9</td>
</tr>
</tbody>
</table>
### 2.2.1 Button – [2.2.1 - Search certificate holder]:

Set `bFound` to false ✓
- Read name of certificate holder from edit field ✓
- Text file: Error handling (try..except OR if File exists)✓
  - Assign, Reset, ✓
  - Show message ✓ and terminate application

Loop through text file ✓
- Read line ✓
  - If line contains name of certificate holder ✓
    - Set `found` to true ✓
  - Find position of `;` in line and obtain/delete name of certificate holder from line ✓
  - Find position of `#` in line and copy issueNr from line ✓, convert to integer ✓ (can assign to variable)
  - Extract expiry date ✓ (can assign to variable)
  - Extract the security code ✓ (can assign to variable)

- Instantiate `objDigCert` ✓ with all four arguments ✓
  - (name of certificate holder, expiry date, security code and issueNr)
- Show panel with buttons ✓
- End loop
- Close file

If name of certificate holder is NOT in file (Found false) ✓
- The panel with buttons should not be visible and display suitable message ✓

---

### 2.2.2 Button – [2.2.2 - Display]:

- Clear output area ✓
- Use `toString` method ✓ to display object information ✓

---

### 2.2.3 Button – [2.2.3 – Test if certificate has expired]:

- Test if certificate has expired using the `hasExpired` function ✓
  - Ask if the digital certificate must be re-issued using an input box or a message dialog box with the correct number of parameters ✓
    - If digital certificate must be re-issued ✓
      - Call methods using the object name ✓
        - `increaseIssueNr` ✓
        - `generateSecurityCode` ✓
        - `resetExpiryDate`
  - Else ✓
    - Display message to indicate that the digital certificate has not expired ✓

- Use `toString` method to display object ✓ or by calling button `btn2_2_2`.

---

**TOTAL SECTION B**

| 2.2.1 | 19 |
| 2.2.2 | 3 |
| 2.2.3 | 8 |

**TOTAL SECTION B**

58
### ANNEXURE C

### SECTION C

#### QUESTION 3: MARKING GRID – PROBLEM SOLVING PROGRAMMING

<table>
<thead>
<tr>
<th>QUESTION</th>
<th>DESCRIPTION</th>
<th>MAX. MARKS</th>
<th>LEARNER’S MARKS</th>
</tr>
</thead>
</table>
| 3.1      | **Button [3.1 - Sales information]**  
Heading: Join 'Department' to week number ✓  
Display heading ✓  
Loop for each department ✓ \{iRow\} (1 to 8)  
Set line variable to department name ✓  
Loop for each week ✓ \{iCol\} (1 to 6)  
Join sales figure from 2D array to line ✓  
Display line variable ✓  
Accept hard coding if reference is made to the index values of the array.  
Subtract two marks will if the String grid is used:  
Set line variable to department name – 1 mark  
Join sales figure – 1 mark | 7 |
| 3.2      | **Button [3.2 - Display underperforming departments]**  
Display the heading ✓  
Loop for each week ✓ \{column\} (1 to 6)  
Initialize sum to zero ✓  
Loop for each department ✓ (1 to 8) nested loop ✓  
Increment the sum ✓ with the sales figure ✓  
Average = sum / 8 (number of departments) ✓  
Display week's heading with average sales figure ✓  
Loop for each department ✓ \{row\} (1 to 8)  
Check IF sales figure ✓ is less than average ✓  
Display department name ✓  
and sales figure in currency ✓ | 14 |
### 3.3 Button [3.3 - New week]

Assigning the file ✓ with the word ‘Week’ and correct week number ✓ (accept Week 1)

Rewrite command ✓
Loop from 1 to number of departments ✓
  Write department name ✓ and sales figure to file ✓
Close file command ✓

Increase start week variable ✓ or any other suitable variable
Loop from 1 to number of departments ✓ (1 to 8)
  Loop from 1 to number of weeks – 1 ✓ (5 times)
    Move sales figures ✓ one position to the left ✓

Loop from 1 to number of departments ✓ (1 to 8)
  Populate arrSales in column 6 ✓ with random data in the range 500 – 5000 ✓

Display updated arrays ✓

Accept any way of generating data in the given range.
Accept integer or real.
Accept random values from 499 to 5001 (inclusive)

| TOTAL SECTION C | 37 |
### 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>55</td>
</tr>
<tr>
<td>MAX. MARKS</td>
<td>55</td>
</tr>
<tr>
<td>LEARNER’S MARKS</td>
<td></td>
</tr>
</tbody>
</table>

Please turn over
ANNEXURE D: SOLUTION FOR QUESTION 1

unit Question1_U;

interface

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

type
  TfrmQuestion1 = class(TForm)
    btnClose: TBitBtn;
    PageControl1: TPageControl;
    tabQues1_1: TTabSheet;
    pnlQ1_1: TPanel;
    tabQues1_2: TTabSheet;
    btnQ1_2_2: TButton;
    tabQues1_4: TTabSheet;
    tabQues1_5: TTabSheet;
    pnlQ1_5_1: TPanel;
    btnQ1_5_1: TButton;
    redQ1_5_1: TRichEdit;
    pnlQ1_5_2: TPanel;
    btnQ1_5_2: TButton;
    redQ1_5_2: TRichEdit;
    tabQues1_3: TTabSheet;
    imgCakePic: TImage;
    lblNumCakes: TLabel;
    btnQ1_3: TButton;
    cmbNumCakes: TComboBox;
    Panel4: TPanel;
    edtNum1: TEdit;
    edtNum2: TEdit;
    lblNumber1: TLabel;
    lblNumber2: TLabel;
    btnQ1_2_1: TButton;
    edtQ1_2_1: TEdit;
    Panel5: TPanel;
    edtWord1: TEdit;
    edtWord2: TEdit;
    lblWord1: TLabel;
    lblWord2: TLabel;
    Panel1: TPanel;
    rgpQ1_4_1: TRadioGroup;
    pnlQ1_4: TPanel;
    edtPassword: TEdit;
    lblPassword: TLabel;
    btnQ1_4_2: TButton;
    pnlHeadingQ1_3: TPanel;
    lblCost: TLabel;
    edtCost: TEdit;
    edtSugarPacks: TEdit;
    lblSugarPacks: TLabel;
    lblSugarInGrams: TLabel;
    edtSugarInGrams: TEdit;
    btnQ1_4_3: TButton;
  end;

implementation

end.
procedure btnQ1_2_2Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure btnQ1_3Click(Sender: TObject);
procedure btnQ1_5_2Click(Sender: TObject);
procedure btnQ1_5_1Click(Sender: TObject);
procedure cmbNumCakesChange(Sender: TObject);
procedure btnQ1_2_1Click(Sender: TObject);
procedure rgpQ1_4_1Click(Sender: TObject);
procedure btnQ1_4_2Click(Sender: TObject);
procedure btnQ1_4_3Click(Sender: TObject);

private
{ Private declarations }

public
{ Public declarations }

end;

var
  frmQuestion1: TfrmQuestion1;
  iNumCakes: integer;
  sPassword: String;

implementation

{$R *.dfm}

//======================================================================
// Question 1.1  (3 marks)
//======================================================================
procedure TfrmQuestion1.FormCreate(Sender: TObject);
begin
  pnlQ1_4.Hide;
  btnQ1_4_3.Enabled := false;

  pnlQ1_1.Color := clLime;
  pnlQ1_1.Font.Size := 15;
  pnlQ1_1.Caption := 'IT is FUN!';
end;

//======================================================================
// Question 1.2.1 (4 marks)
//======================================================================
procedure TfrmQuestion1.btnQ1_2_1Click(Sender: TObject);
var
  iNum1, iNum2: integer;
begin
  iNum1 := StrToInt(edtNum1.Text);
  iNum2 := StrToInt(edtNum2.Text);
  if (iNum1 > iNum2) then
    edtQ1_2_1.Text := IntToStr(iNum1)
  else
    if (iNum2 > iNum1) then
      edtQ1_2_1.Text := IntToStr(iNum2)
    else
      edtQ1_2_1.Text := 'Equal';
OR
if (iNum1 = iNum2) then
  edtQ1_2_1.Text := 'Equal'
else
  edtQ1_2_1.Text := intToStr(Max(iNum1,iNum2));
end;

// Question 1.2.2 (5 marks)
//------------------------------------------------------------------------------
procedure TfrmQuestion1.btnQ1_2_2Click(Sender: TObject);
// Provided code
var
  sWord1, sWord2: String;
  sTempWord: String;
begin
  sWord1 := edtWord1.Text;
  sWord2 := edtWord2.Text;
  sTempWord := sWord1;
  sWord1 := sWord2;
  sWord2 := sTempWord;

  edtWord1.Text := sWord1;
  edtWord2.Text := sWord2;
end;

// Question 1.3.1 (5 Marks)
//------------------------------------------------------------------------------
procedure TfrmQuestion1.cmbNumCakesChange(Sender: TObject);
// Provided code
const
  PRICE = 159.50;
var
  rCost: Real;
begin
  iNumCakes := cmbNumCakes.ItemIndex + 1;
  imgCakePic.Picture.LoadFromFile('Pict' + IntToStr(iNumCakes) + '.PNG');
  rCost := iNumCakes * PRICE;
  edtCost.Text := FloatToStrF(rCost, ffCurrency, 6, 2);
end;

// Question 1.3.2 (5 marks)
//------------------------------------------------------------------------------
procedure TfrmQuestion1.btnQ1_3Click(Sender: TObject);
// Provided code
const
  SUGAR = 375;
var
  iSugarGrams, iSugarPacks: integer;
begin
  iSugarGrams := iNumCakes * SUGAR;
  edtSugarInGrams.Text := IntToStr(iSugarGrams);
  iSugarPacks := Ceil (iSugarGrams / 1000);
  edtSugarPacks.Text := IntToStr(iSugarPacks);
end;
// Question 1.4.1 (4 marks)
// Question 1.4.1 (4 marks)
procedure TfrmQuestion1.rgpQ1_4_1Click(Sender: TObject);
begin
    if (rgpQ1_4_1.ItemIndex = 0) OR (rgpQ1_4_1.ItemIndex = 2) then
        pnlQ1_4.Show
    else
        pnlQ1_4.Hide;
end;

// Question 1.4.2 (11 marks)
// Question 1.4.2 (11 marks)
procedure TfrmQuestion1.btnQ1_4_2Click(Sender: TObject);
var
    i, iCountChar: Integer;
    bValid: Boolean;
begin
    bValid := false;
    iCountChar := 0;
    sPassword := edtPassword.Text;
    if length(sPassword) >= 6 then
        begin
            if sPassword[1] in ['A' .. 'Z'] then
                for i := 2 to length(sPassword) do
                    if sPassword[i] in ['$', '@', '#', '&'] then
                        Inc(iCountChar);
            if iCountChar >= 2 then
                ShowMessage('Valid Password');
                btnQ1_4_3.Enabled := true;
                bValid := true;
        end;
    if (bValid = false) then
        begin
            ShowMessage('Invalid Password');
            edtPassword.Text := '';
        end;
end;

// Question 1.4.3 (5 marks)
// Question 1.4.3 (5 marks)
procedure TfrmQuestion1.btnQ1_4_3Click(Sender: TObject);
begin
    if sPassword[1] = 'Z' then
        sPassword[1] := 'A'
    else
        sPassword[1] := char(ord(sPassword[1])+1);
    edtPassword.Text := sPassword;
end;
// Question 1.5.1  (6 marks)
procedure TfrmQuestion1.btnQ1_5_1Click(Sender: TObject);
var
   iNum: integer;
   rSquareRoot: Real;
begin
   redQ1_5_1.Clear;
   iNum := StrToInt(InputBox('Perfect Square', 'Enter number', ''));
   rSquareRoot := Sqrt(iNum);
   if rSquareRoot = trunc(rSquareRoot) then
      redQ1_5_1.Lines.Add(IntToStr(iNum) + ' is a perfect square.');
   else
      redQ1_5_1.Lines.Add(IntToStr(iNum) + ' is not a perfect square.');</n
// Question 1.5.2  (7 marks)
procedure TfrmQuestion1.btnQ1_5_2Click(Sender: TObject);
// Provided code
const
   MULTIPLIER = 3;
var
   iSum, iNum: integer;
   sOutput: String;
begin
   redQ1_5_2.Clear;
   sOutput := '';
   iSum := 0;
   iNum := 1;
   repeat
      sOutput := sOutput + IntToStr(iNum) + '  ';  
      iSum := iSum + iNum;
      iNum := iNum * MULTIPLIER;
   until iSum > 1000;
   redQ1_5_2.Lines.Add(sOutput);
end;
ANNEXURE E: SOLUTION FOR QUESTION 2

OBJECT CLASS:

unit DCertificate_U;

interface

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

type
  TDigCertificate = class(TObject)
    private
      fCertHolder: String;
      fExpiryDate: String;
      fSecurityCode: String;
      fIssueNr: Integer;
    public
      constructor Create(sCertHolder, sExpdate: String; sCode: String;
        iIssueNr: Integer);
      procedure increaseIssueNr;
      procedure resetExpiryDate;
      function hasExpired: boolean;
      procedure generateSecurityCode;
      function toString: String;
    end;

implementation

var
  sSysdate: String;

//======================================================================
// Question 2.1.1  (4 marks)
//======================================================================
constructor TDigCertificate.Create(sCertHolder, sExpdate: String; sCode: String;
  iIssueNr: Integer);
begin
  fCertHolder := sCertHolder;
  fExpiryDate := sExpdate;
  fSecurityCode := sCode;
  fIssueNr := iIssueNr;
end;

//======================================================================
// Question 2.1.2  (2 marks)
//======================================================================
procedure TDigCertificate.increaseIssueNr;
begin
  inc(fIssueNr);
end;
Question 2.1.3  (4 marks)

procedure TDigCertificate.resetExpiryDate;
var
  sYear: String;
  iYear: Integer;
begin
  // Provided code
  ShortDateFormat := ('dd/mm/yyyy');
  sSysdate := FormatDateTime('dd/mm/yyyy', Date);

  sYear := Copy(sSysdate, 7, 4);
  iYear := StrToInt(sYear) + 1;
  fExpiryDate := Copy(sSysdate, 1, 6) + IntToStr(iYear);
  // OR
  // fExpiryDate := DateToStr(incYear(StrToDate(sSysDate), 1));
end;

Question 2.1.4  (5 marks)

function TDigCertificate.hasExpired: boolean;
begin
  // Provided code
  sSysdate := FormatDateTime('dd/mm/yyyy', Date);
  ShowMessage(sSysdate);
  if StrToDate(fExpiryDate) < StrToDate(sSysdate) then
    Result := true;
  else
    Result := false;
end;

Question 2.1.5  (10 marks)

procedure TDigCertificate.generateSecurityCode;
var
  iRNum, I: Integer;
  sSecurityCode: String;
  sChars: String;
  // iRNum: Integer;
begin
  sSecurityCode := '';
  sChars := '0123456789ABCDEF';
  for I := 1 to 14 do
    if (I mod 3 = 0) then
      sSecurityCode := sSecurityCode + ':'
    else
      begin
        iRNum := random(16) + 1;
        sSecurityCode := sSecurityCode + sChars[iRNum];
      end;
  fSecurityCode := sSecurityCode;
// Alternative solution
{ for I := 1 to 10 do
    begin
    iRNum := Random(16);
    case iRNum of
        0 .. 9:  sChar := IntToStr(iRNum);
        10:      sChar := 'A';
        11:      sChar := 'B';
        12:      sChar := 'C';
        13:      sChar := 'D';
        14:      sChar := 'E';
        15:      sChar := 'F';
    end;

    // OR   sChar := IntToHex(iRNum,1);

    if (I mod 2 = 0) AND NOT(I = 10) then
        sSecurityCode := sSecurityCode + sChar + ':'
    else
        sSecurityCode := sSecurityCode + sChar;
    end; }

====================================================================
// Question 2.1.6   (3 marks)
====================================================================
function TDigCertificate.toString;
var
    sOut: String;
begin
    sOut := 'Digital certificate information:' + #13#13;
    sOut := sOut + 'Certificate holder: ' + #9 + fCertHolder + #13#13;
    sOut := sOut + 'Expiry date: ' + #9 + fExpiryDate + #13#13;
    sOut := sOut + 'Security code: ' + #9 + fSecurityCode + #13#13;
    sOut := sOut + 'Issue number: ' + #9 + IntToStr(fIssueNr);
    result := sOut;
end;
end.
MAIN FORM UNIT: QUESTION2_U.PAS

unit Question2_U;

interface

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

type
  TfrmQuestion2 = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    Label1: TLabel;
    Panel3: TPanel;
    Panel4: TPanel;
    btnQ2_2_1: TButton;
    btnQ2_2_2: TButton;
    redOutput: TRichEdit;
    btnClose: TBitBtn;
    btnReset: TBitBtn;
    pnlDate: TPanel;
    edtCertificateHolder: TEdit;
    pnlQ2_Buttons: TPanel;
    btnQ2_2_3: TButton;
  private
    { Private declarations }
    objDigCert: TDigCertificate;
  public
    { Public declarations }
  end;

var
  frmQuestion2: TfrmQuestion2;
  sSysDate: String;

implementation

{$R *.dfm}

// Question 2.2.1 (19 marks)
procedure TfrmQuestion2.btnQ2_2_1Click(Sender: TObject);
var
  tFile: TextFile;
  sLine, sCertHolder, sHolder, sExpDate, sCode: String;
  iIssueNr, iPos, iPosHash, iPosDash: Integer;
  bFound: boolean;
begin
iIssueNr := 0;
bFound := false;
sCertHolder := edtCertificateHolder.Text;

AssignFile(tFile, 'DigitalCertificates.txt');
try
  reset(tFile);
except
  ShowMessage('File not found');
  EXIT;
end;

while NOT eof(tFile) and NOT(bFound) do
begin
  readln(tFile, sLine);
iPos := pos(';', sLine);
sHolder := copy(sLine, 1, iPos - 1);
if (sCertHolder = sHolder) then
begin
  bFound := true;
  Delete(sLine, 1, iPos);
iPosHash := pos('#', sLine);
iIssueNr := strToInt(copy(sLine, 1, iPosHash - 1));
delete(sLine, 1, iPosHash);
iPosHash := pos('#', sLine);
sExpDate := copy(sLine, 1, iPosHash - 1);
sCode :=  copy(sLine, iPosHash + 1);
end;
end;

if bFound then
begin
  objDigCert := TDigCertificate.Create(sCertHolder, sExpDate, sCode, iIssueNr);
  pnlQ2_Buttons.Visible := true;
end
else
begin
  pnlQ2_Buttons.Visible := false;
  ShowMessage(sCertHolder + ' was not found');
end;
end;

// =====================================================================
// Question 2.2.2               (3 marks)
// Question 2.2.2               (3 marks)
procedure TfrmQuestion2.btnQ2_2_2Click(Sender: TObject);
begin
  redOutput.Lines.Clear;
  redOutput.Lines.Add(objDigCert.toString);
end;
procedure TfrmQuestion2.btnQ2_2_3Click(Sender: TObject);
var
  sAnsw: String;
begin
  if NOT(objDigCert.hasExpired) then
    begin
      ShowMessage('Digital certificate has not expired');
    end
  else
    begin
      sAnsw := InputBox('Validation', 'Digital certificate has expired.'+
        #13 + 'Do you want to renew your digital certificate (Y/N)?', 'Y');
      if UpperCase(sAnsw) = 'Y' then
        begin
          objDigCert.resetExpiryDate;
          objDigCert.increaseIssueNr;
          objDigCert.generateSecurityCode;
        end;
    end;
  btnQ2_2_2.Click;
end;

procedure TfrmQuestion2.FormCreate(Sender: TObject);
begin
  ShortDateFormat := ('dd/mm/yyyy');
  DateSeparator := '/';
  sSysDate := FormatDateTime('dd/mm/yyyy', Date);
  pnlDate.Caption := sSysDate;
  redOutput.Paragraph.TabCount := 1;
  redOutput.Paragraph.Tab[0] := 120;
  pnlQ2_Buttons.Visible := false;
//pnlDate.Caption := '17/10/2017'; //Set date for test purposes
end;

procedure TfrmQuestion2.btnResetClick(Sender: TObject);
begin
  pnlQ2_Buttons.Visible := false;
  edtCertificateHolder.Clear;
  edtCertificateHolder.SetFocus;
  redOutput.Clear;
end;

// Provided code
// =====================================================================
ANNEXURE F: SOLUTION FOR QUESTION 3

unit Question3_U;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, Buttons, ExtCtrls, Grids, ComCtrls;
type
  TfrmQuestion3 = class(TForm)
    pnlBtn: TPanel;
    btnClose: TBitBtn;
    btnQues31: TButton;
    btnQues33: TButton;
    btnQues32: TButton;
    redQues3: TRichEdit;
    pnlHeading: TPanel;
  procedure FormCreate(Sender: TObject);
  procedure btnQues31Click(Sender: TObject);
  procedure btnQues32Click(Sender: TObject);
  procedure Display(iStartWeek: integer);
  procedure WriteToFile(iWeekNumber: integer);
  procedure btnQues33Click(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;

var
  frmQuestion3: TfrmQuestion3;

implementation
{$R *.dfm}
{$R+}
//======================================================================
// Provided code
//======================================================================
var
  arrDepartments : array[1..8] of String = ('PCs & Laptops', 'Tablets & eReaders', 'Software',
                                           'Printers, Toners and Ink', 'Cellphones', 'Games & Drones ',
                                           'Network equipment', 'Accessories');

  arrSales: array [1..8, 1..6] of Real = (935.89, 965.99, 4056.77, 5023.89, 3802.66, 1146.98),
                                           (2667.78, 2491.78, 1989.65, 2647.88, 1601.56, 1921.99),
                                           (6702.45, 4271.56, 3424.45, 3924.55, 3085.45, 3359.77),
                                           (6662.34, 6658.45, 8075.43, 2360.66, 2635.44, 7365.69),
                                           (16405.33, 9741.37, 13381.56, 18969.76, 8604.55, 20207.56),
                                           (10515.29, 7582.66, 9856.56, 7537.68, 9115.67, 8401.55),
                                           (7590.99, 9212.65, 9070.98, 6439.99, 7984.88, 8767.45),
                                           (9220.65, 8097.99, 10067.44, 9960.87, 10109.56, 6571.66));

  iStartWeek: Integer = 1;
Question 3.1  (7 marks)

procedure TfrmQuestion3.btnQ3_1Click(Sender: TObject);
var
    iRow, iCol : Integer;
    sLine : String;
begin
    Display(iStartWeek); // Display headings
end;

Procedure Display

procedure TfrmQuestion3.Display(iStartWeek: Integer);
var
    sLine: String;
    iRow, iCol: Integer;
    I: Integer;
begin
    sLine := 'Department' + #9;
    for I := iStartWeek to iStartWeek + 5 do
        sLine := sLine + 'Week ' + IntToStr(I) + #9;
    redQ3.Lines.Add(sLine);
    for iRow := 1 to Length(arrDepartments) do
        begin
            sLine := arrDepartments[iRow] + #9;
            for iCol := 1 to 6 do
                begin
                    sLine := sLine + FloatToStrF(arrSales[iRow, iCol], ffCurrency, 8, 2) + #9;
                end;
            redQ3.Lines.Add(sLine);
        end;
end;

Question 3.2  (14 marks)

procedure TfrmQuestion3.btnQ3_2Click(Sender: TObject);

function AvgForWeekX(WeekNr: Integer): Real;
// Local function
var
    iRow: Integer;
    rSum, rAvg: Real;
begin
    rSum := 0;
    for iRow := 1 to Length(arrDepartments) do
        rSum := rSum + arrSales[iRow, WeekNr];
    rAvg := rSum / Length(arrDepartments);
    Result := rAvg;
end;
begin
//Display the underperforming departments per week.
redQ3.Clear;

redQ3.Lines.Add('Underperforming departments per week:');
for iCol := 1 to 6 do
begin
    rAvg := AvgForWeekX(iCol);
    redQ3.Lines.Add('Week ' + IntToStr(iCol) + ': Avg sales: ' + FloatToStrF(rAvg, ffCurrency, 8, 2));
    for iRow := 1 to Length(arrDepartments) do
    begin
        if arrSales[iRow, iCol] < rAvg then
            redQ3.Lines.Add(arrDepartments[iRow] + #9 + FloatToStrF(arrSales[iRow, iCol], ffCurrency, 8, 2));
    end;
end;
redQ3.Lines.Add(' ');
end;

 //======================================================================
// Question 3.3               (16 marks)
//======================================================================
procedure TfrmQuestion3.btnQ3_3Click(Sender: TObject);
var
    iRow, iCol: Integer;
begin
    WriteToFile(iStartWeek);
    Inc(iStartWeek);
    for iRow := 1 to Length(arrDepartments) do
        for iCol := 1 to 5 do
            arrSales[iRow, iCol] := arrSales[iRow, iCol + 1];
    for iRow := 1 to Length(arrDepartments) do
        arrSales[iRow, 6] := random(4501) + 500 + random;
    redQ3.Clear;
    Display(iStartWeek);
end;

procedure TfrmQuestion3.WriteToFile(iWeekNumber: integer);
var
    tFile: TextFile;
    iRow : Integer;
begin
    AssignFile(tFile, 'Week ' + IntToStr(iWeekNumber) + '.txt');
    Rewrite(tFile);
    for iRow := 1 to Length(arrDepartments) do
        Writeln(tFile, arrDepartments[iRow] + ': ' + FloatToStrF(arrVerkope[iRow, 1], ffCurrency, 6, 2));
    CloseFile(tFile);
end;
procedure TfrmQuestion3.FormCreate(Sender: TObject);
var
  iCol : Integer;
begin
  //***  PROVIDED CODE >> DO NOT CHANGE !!!  ***
  {$Region Provided Code}
  //Setup the columns in the richEdit
  frmQuestion3.Width := 780;
  redQues3.Paragraph.TabCount := 6;
  redQues3.Paragraph.Tab[0]   := 175;
  for iCol := 1 to 6 do
    redQues3.Paragraph.Tab[iCol] := 175 + (65*iCol);
  CurrencyString := 'R ';
  ThousandSeparator := ' ';
  {$EndRegion}
end;
end.