basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

SENIOR CERTIFICATE EXAMINATIONS/
NATIONAL SENIOR CERTIFICATE EXAMINATIONS

INFORMATION TECHNOLOGY P1

2021

MARKING GUIDELINES

MARKS: 150

These marking guidelines consist of 23 pages.

GENERAL INFORMATION:

- These marking guidelines are to be used as the basis for the marking session. They were prepared for use by markers. All 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 paper was not followed or the requirements of the question was not met

- **Annexures A, B, C** and **D** (pages 3 to 10) include the marking grid for each question.

- **Annexures E, F, G** and **H** (pages 11 to 23) contain examples of solutions for QUESTIONS 1 to 4 in programming code.

- Copies of **Annexures A, B, C** and **D** (pages 3 to 10) should be made for each learner and completed during the marking session.

ANNEXURE A

QUESTION 1: MARKING GRID – GENERAL PROGRAMMING SKILLS

CENTRE NUMBER:	**EXAMINATION NUMBER:**		
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
1.1	**Button [1.1 – Calculate thickness of slices]** Declare an integer variable and a real variable ✓ Stretch image with code ✓ Extract number of slices from spin edit ✓ Thickness of slice = (242 / ✓ number of slices) ✓ Display on lblQ1_1, thickness of slices ✓ formatted to two decimal places ✓	**7**	
1.2	**Button [1.2 – Calculate change]** Set constant BREAD_PRICE = 12.90 ✓ Declare two real values ✓ Extract amount offered from edit box ✓ converted to float/real ✓ If Amount offered >= BREAD_PRICE ✓ Change = Amount offered ✓ - BREAD_PRICE ✓ Display on panel Change ✓ as Currency ✓ else ✓ Display 'Insufficient amount offered' on panel✓ **Alternative for If:** If Change >= 0 **Alternative for else:** If Amount offered < BREAD_PRICE	**11**	
1.3	**Button [1.3 – Multiples of 10]** Set Counter for multiples of 10 to 0 ✓ Loop ✓ 10 times ✓ Generate random✓ number in the range 50 to 100 ✓ Display random value ✓ Test if random value ✓ is a multiple of 10 ✓ Increment Counter ✓ Display Counter with message ✓	**10**	

1.4	**Button [1.4 – Hidden security code]** Find the position of 't' ✓ While ✓ (length security code < 8) ✓ AND (Pos 't' > 0) ✓ Test if character at position -1 ✓ is not ' ' ✓ Add character at position -1✓ to security code ✓ Delete characters from paragraph up to position of 't' ✓ Find position of 't' ✓ Display security code ✓in uppercase ✓ **CONCEPTS:** Loop (1) from 2 (1) through provided paragraph (1) Test security code length < 8 (1) Test character at index (1) = 't' (1) Test if character at index -1 (1) is not space (1) Add character at index -1 (1) to security code (1) Display security code (1) in uppercase (1) **NOTE**: Loop can start at 1 up to length with different code for testing	**12**	
	TOTAL SECTION A:	**40**	

ANNEXURE B

QUESTION 2: MARKING GRID – DATABASE PROGRAMMING

CENTRE NUMBER:	EXAMINATION NUMBER:

QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
2.1	**SQL statements**		

2.1.1	**Button [2.1.1 – List of shops]**
	`SELECT * OR name all fields ✓`
	`FROM tblShops ✓`
	`ORDER BY ShopSize DESC ✓`
		3	

2.1.2	**Button [2.1.2 – Manager number]**
	`SELECT ShopName ✓`
	`FROM tblShops ✓`
	`WHERE ManagerNumber = ' + sManagerNum ✓`
		3	

2.1.3	**Button [2.1.3 – Rental > R50 000]**
	`SELECT ShopName, ShopSize,`
	`Format(65 * ShopSize ✓,"Currency"✓)`
	`AS Rental ✓`
	`FROM tblShops`
	`WHERE 65 * ShopSize ✓ > 50000 ✓`
	
	`Alternative:`
	`GROUP BY Shopname,Shopsize HAVING (ShopSize * 65 > 50000)`
		5	

2.1.4	**Button [2.1.4 – Turnover per manager]**
	`SELECT ManagerName, ManagerSurname ✓`
	`FORMAT(SUM(TurnOver) ✓, "Currency")`
	`AS TotalTurnover ✓`
	`FROM tblShops S, tblManagers M ✓`
	`WHERE S.ManagerNumber = M.ManagerNumber ✓`
	`Correct aliases or table names`
	`GROUP BY ManagerName, ManagerSurname ✓`
		6	

2.1.5	**Button [2.1.5 – Update shop name]**
	`UPDATE tblShops ✓`
	`SET ShopName = "TeenDream" ✓`
	`WHERE ShopName = "Jeans 4U" ✓`
		3	

	Subtotal:	**20**	

QUESTION 2: MARKING GRID (CONT.)

2.2	**DATABASE MANIPULATION**		
2.2.1	**Button [2.2.1 – Number of small shops]** Move to the first record in tblShops ✓ Set a counter to 0 ✓ Loop through tblShops table ✓ Test if Shop Size smaller than 300 then ✓ Add 1 to the counter ✓ Move to the next record in tblShops table ✓ Display the total of small shops (counter) ✓	**7**	

2.2.2	**Button [2.2.2 – Display manager]**
	
	Set a flag bFound to false ✓
	Start at the first record of tblShops ✓
	Loop through tblShops table ✓
	Test if ShopName in the table equals the name of shop extracted from the inputbox ✓
	Set the flag bFound to True ✓
	Assign the manager number in tblShops table to the a variable (iManager) ✓
	Move to the next record in tblShops table ✓
	end (loop)
	
	Test if bFound is true ✓
	Start at the first record of tblManagers ✓
	Loop through tblManagers table (with correct .next) ✓
	if ManagerNumber in the tblManagers table is equal to iManager ✓ then
	Display the manager name, surname and contact number ✓
	Move to next record in tblManagers
	else
	Display a message that the shop is not in this mall ✓
		13	

	Subtotal:	**20**	
	TOTAL SECTION B:	**40**	

ANNEXURE C

QUESTION 3: MARKING GRID – OBJECT-ORIENTED PROGRAMMING

CENTRE NUMBER:	**EXAMINATION NUMBER:**		
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
3.1.1	**Constructor method:** Heading (method declaration) with three parameter values ✓ of correct data types ✓ Assign name parameter value to correct attribute ✓ Assign balance and voucher number parameter values to correct attributes ✓	**4**	
3.1.2	**getBalance function:** Function heading/declaration with real value as return data type ✓ fBalance assigned to result or function name ✓ **getVoucherNumber function:** Function heading/declaration with integer value as return data type ✓ fVoucherNumber assigned to result✓ or assign fVoucherNumber to function name: *getVoucherNumber := fVoucherNumber;*	**4**	
3.1.3	**isSufficient function:** Function heading/declaration and boolean value as return data type ✓ with parameter of real/float data type ✓ Test if parameter value <= balance attribute ✓ result = true ✓ Else result = false ✓ **Alternative code for test** Set result to fBalance >= parameter value (3)	**5**	
3.1.4	**updateBalance procedure:** Procedure heading/declaration receiving parameter of real/float data type ✓ Balance = Balance ✓ – parameter value ✓	**3**	
3.1.5	**toString method:** Three text labels (Voucher number, Customer name, Available balance) ✓ and three attribute values ✓ in correct format ✓ Return the string ✓	**4**	
	Subtotal: Object class	**20**	

QUESTION 3: MARKING GRID (CONT.)

QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
3.2.1	**Button [Purchase]** Extract gift voucher number and name from edit boxes ✓ Extract amount from combobox ✓ *Instantiate the objGiftVoucher object:* objGiftVoucher:= ✓TGiftVoucher.create ✓ Use three arguments with correct data type and in correct order✓ Use toString method to display gift voucher information in richedit component ✓	**6**	
3.2.2(a)	**Button [Display balance]** Extract gift voucher number using edit box ✓ Test if it is the correct gift voucher using getVoucherNumber method✓ Display balance on panel using getBalance method ✓ in currency format ✓ Set btnQ3_2_2_b to enabled ✓	**5**	
3.2.2(b)	**Button [Use gift voucher]** Extract purchase amount from edit box and convert ✓ If objGiftVoucher.isSufficient(rPurchase) ✓ Update balance using the updateBalance method ✓ Display 'Voucher successfully used' on the label ✓ else Calculate outstanding amount ✓ Display message and outstanding amount formatted as currency on the label ✓ Update the balance using updateBalance✓ with the getBalance method as argument ✓ Display the updated balance in the panel ✓	**9**	
	Subtotal Form class:	**20**	
	TOTAL SECTION C:	**40**	

ANNEXURE D

QUESTION 4: MARKING GRID – PROBLEM-SOLVING PROGRAMMING

CENTRE NUMBER:	**EXAMINATION NUMBER:**		
SECTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
4.1	**Button [4.1 – Populate and display array]** Display headings✓ Extract selected mall name from radio button✓ Assign text file using correct mall name.txt✓ Reset text file✓ Set row counter to 0 ✓ Loop to end of text file ✓ Increase row counter ✓ Read line from text file✓ Store line in array at correct position✓ *Compile output string:* Row counter and full stop ✓ Add spaces at correct position✓ Until length of line is 25 (or suitable) characters✓ Add shops line ✓ Display output string✓ and length of line✓ Close file **CONCEPTS:** Get text file name (1) Assign and reset (2) Read content of text file (2) Managing index of array (2) Save to an array (1) Compile output string • Start with line number (1) • Add correct number of spaces (2) • Add the shops line (1) Display heading and shops per level (3)	**15**	

QUESTION 4: MARKING GRID (CONT.)

4.2	**Button [4.2 – Count type of shop per level]** Enter shop type using an inputbox ✓ Structure to test for valid shop letters✓ Test if letter✓ is NOT valid✓ Display error message ✓and list of correct letters Else (if letter IS valid) Display heading including type of shop from array✓ Loop through levels✓ Initialize counter for shops per level to 0✓ Set output string to level counter ✓+ full stop Loop through the length of string value in arrShops✓ - nested loop/inner loop ✓ Test if letter equals entered shop type✓ Increase counter✓ Add shop counter to display ✓ Display results for level ✓ **CONCEPTS:** Enter shop type (1) Test if entered shop type is valid (1) using an appropriate structure/method (2) display message for incorrect shop type (1) Display heading including shop type (1) Count the number of shops at every level Using nested loops (6) Initialize a string with level counter (1) Add shop counter (1) and display in outer loop (1)	**15**	
	TOTAL SECTION D:	**30**	
	GRAND TOTAL:	**150**	

SUMMARY OF LEARNER'S MARKS:

CENTRE NUMBER:		**LEARNER'S EXAMINATION NUMBER:**			
	SECTION A	**SECTION B**	**SECTION C**	**SECTION D**	
	QUESTION 1	**QUESTION 2**	**QUESTION 3**	**QUESTION 4**	**GRAND TOTAL**
MAX. MARKS	**40**	**40**	**40**	**30**	**150**
LEARNER'S MARKS					

ANNEXURE E: SOLUTION FOR QUESTION 1

```
// ========================================================================
// Question 1.1          7 marks
// ========================================================================
procedure TfrmQuestion1.btnQ1_1Click(Sender: TObject);
var
  iNumSlice: integer;
  rThickness: real;
begin
  // Question 1.1
  imgQ1_1.Stretch := true;
  iNumSlice := spnQ1_1.Value;
  rThickness := (242 / iNumSlice);
  lblQ1_1.Caption := FloatToStrF(rThickness ,ffFixed,6,2);
end;
// ========================================================================
// Question 1.2          11 marks
// ========================================================================
procedure TfrmQuestion1.btnQ1_2Click(Sender: TObject);
const
  BREAD_PRICE = 12.90;
var
  rAmount, rChange: real;
begin
  // Question 1.2
  rAmount := StrToFloat(edtQ1_2.Text);
  rChange := rAmount - BREAD_PRICE;
  if rAmount >= BREAD_PRICE then
    pnlQ1_2.Caption := 'Change: ' + FloatToStrF(rChange,ffCurrency,8,2)
  else
     pnlQ1_2.Caption := 'Insufficient amount offered';
  // Alternative: if with condition instead of else:
  // if rAmount < BREAD_PRICE then
end;
// ========================================================================
// Question 1.3          10 marks
// ========================================================================
procedure TfrmQuestion1.btnQ1_3Click(Sender: TObject);
var
  iLoopCounter, iCntMultiple10: integer;
  iRandomValue: integer;
begin
  // Question 1.3
  redQ1_3.Clear;
  iCntMultiple10 := 0;
  for iLoopCounter := 1 to 10 do
  begin
    iRandomValue := RandomRange(50, 101);  // or alternative ways
    redQ1_3.Lines.Add(IntToStr(iRandomValue));
    if iRandomValue MOD 10 = 0 then
    begin
      inc(iCntMultiple10);
    end;
  end;
```

```
  redQ1_3.Lines.Add(#13 + 'Number of multiples of 10: ' +
                IntToStr(iCntMultiple10));
end;
// ============================================================================
// Question 1.4          12 marks
// ============================================================================
procedure TfrmQuestion1.btnQ1_4Click(Sender: TObject);
var
  sParagraph: string;
  sSecurityCode: string;
  iPos: integer;
  // I: integer;    // required for alternative

begin
  // Question 1.4

  // Provided code
   sParagraph :=
   sParagraph :=  'I am not lazy, I am just very relaxed. He who laughs
     last did not get the joke. When nothing is going right, go left. '
     + 'I love school when it is vacation. I put the "Pro" in
     procrastinate.';

  //sParagraph := 'My Grade 12 year is always going to be my greatest as
it provides me with the most opportunities.';

  sSecurityCode := '';
  iPos := pos('t', sParagraph);
  while (length(sSecurityCode) < 8) AND ( iPos <> 0) do
  begin
    if sParagraph[iPos - 1] <> ' ' then
      sSecurityCode := sSecurityCode + sParagraph[iPos - 1];
    Delete(sParagraph, 1, iPos);
    iPos := pos('t', sParagraph);
  end;
  edtQ1_4.Text := uppercase(sSecurityCode);

// Alternative solution
//  for I := 2 to length(sParagraph) do
//  begin
//    if (sParagraph[I] = 't') AND (sParagraph[I - 1] <> ' ') AND
//      (length(sSecurityCode) < 8) then
//      sSecurityCode := sSecurityCode + sParagraph[I - 1];
//  end;
//  edtQ1_4.Text := uppercase(sSecurityCode);
end;


end.
```

ANNEXURE F: SOLUTION FOR QUESTION 2

```
unit Question2_U;

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;
    pgcDBAdmin: TPageControl;
    tabsQ2SQL: TTabSheet;
    btnQ2_1_1: TBitBtn;
    btnQ2_1_3: TBitBtn;
    btnQ2_1_2: TBitBtn;
    btnQ2_1_4: TBitBtn;
    bmbQ2_1_5: TBitBtn;
    grpQ2_2_1: TGroupBox;
    grpresults: TGroupBox;
    dbgrdSQL: TDBGrid;
    grpQ2_1_3: TGroupBox;
    pnlQDB: TPanel;
    cmbQ2_1_2: TComboBox;
    redQ2_2_1: TRichEdit;
    Label2: TLabel;
    btnQ2_2_1: TButton;
    grpQ2_2_2: TGroupBox;
    btnQ2_2_2: TButton;
    redQ2_2_2: TRichEdit;
    b: TTabSheet;
    grpManagers: TGroupBox;
    grpShops: TGroupBox;
    dbgrdONE: TDBGrid;
    dbgrdMany: TDBGrid;
    procedure bmbRestoreDBClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btnQ2_1_1Click(Sender: TObject);
    procedure btnQ2_1_3Click(Sender: TObject);
    procedure btnQ2_1_2Click(Sender: TObject);
    procedure btnQ2_1_4Click(Sender: TObject);
    procedure bmbQ2_1_5Click(Sender: TObject);
    procedure btnQ2_2_2Click(Sender: TObject);
    procedure btnQ2_2_1Click(Sender: TObject);
  private
  public
  end;

var
  frmDBQuestion2: TfrmDBQuestion2;
  dbCONN: TConnection;
```

```
  // --- Global variables to be used ---
  tblShops, tblManagers: TADOTable;

implementation
{$R *.dfm}
{$R+}
//=========================================================================
// Question 2.1 – SQL section
//=========================================================================

//=========================================================================
//  Question 2.1.1          3 marks
//=========================================================================
     sSQL1 := 'SELECT * FROM tblShops ORDER BY ShopSize DESC';

//=========================================================================
//  Question 2.1.2          3 marks
//=========================================================================
     sSQL2 :=  'SELECT ShopName FROM tblShops WHERE ManagerNumber = '
               + sManagerNum;

//=========================================================================
//  Question 2.1.3          5 marks
//=========================================================================
     sSQL3 :=  'SELECT ShopName, ShopSize,
               Format(65 * ShopSize,"Currency") AS Rental FROM tblShops
               WHERE 65 * ShopSize  > 50000';

//=========================================================================
//  Question 2.1.4          6 marks
//=========================================================================
      sSQL4 := SELECT ManagerName, ManagerSurname,
            format(SUM(turnOver),"Currency") AS [TotalTurnover]
            FROM tblShops S, tblManagers M
            WHERE S.ManagerNumber = M.ManagerNumber
            GROUP BY ManagerName, ManagerSurname';

//=========================================================================
//  Question 2.1.5          3 marks
//=========================================================================
    sSQL5 := 'UPDATE tblShops ' +
             'SET ShopName = " TeenDream" ' +
             'WHERE ShopName = " Jeans 4U" ';
```

```
// ============================================================================
// Question 2.2 -     Delphi code section
// ============================================================================

// ============================================================================
// Question 2.2.1          7 marks
// ============================================================================
   procedure TfrmDBQuestion2.btnQ2_2_1Click(Sender: TObject);
   var
      iCountSmall: Integer;
   begin // Question 2.2.1
   // Enter your code here
   tblShops.First;
   iCountSmall:= 0;
   while not tblShops.Eof do
   begin
    if tblShops['ShopSize'] < 300 then
      inc(iCountSmall);
    tblShops.Next;
   end;
   redQ2_2_1.Lines.Add('Small shops: ' + IntToStr(iCountSmall));

   // Provided code
   dbCONN.setupGrids(dbgrdONE, dbgrdMany, dbgrdSQL);
end;
// ============================================================================
// Question 2.2.2          13 marks
// ============================================================================
   procedure TfrmDBQuestion2.btnQ2_2_2Click(Sender: TObject);
   var
      sShopName: String;
      iManager : Integer;
      bFound   : Boolean;
   begin
      //Provided code
      redQ2_2_1.Clear;
      redQ2_2_2.Paragraph.TabCount := 2;
      redQ2_2_2.Paragraph.Tab[0] := 70;
      redQ2_2_2.Paragraph.Tab[1] := 150;

      redQ2_2_1.Lines.Add('Name' + #9 + 'Surname' + #9 + 'Number');
      sShopName := inputBox('Shop search','Enter shop name to search',
                   'Little Kitchen Grocery Store');

      // Question 2.2.2
      // Enter your code here
      bFound := false;
      tblShops.First;
      while (NOT bFound) and (NOT tblShops.Eof) do
      begin
         if tblShops['ShopName'] = sShopName then
         begin
            bFound := true;
            iManagerNum := tblShops['ManagerNumber']
         end;
```

```
        tblShops.Next;
      end;
      if bFound then
      begin
          tblManagers.First;
         while NOT tblManagers.Eof do
         begin
            if tblManagers['ManagerNumber'] = iManagerNum then
            begin
               redQ2_2_2.Lines.Add(tblManagers['ManagerName'] + #9 +
                  tblManagers['ManagerSurname'] + #9 +
                  tblManagers['ContactNumber']);
            end;
          tblManagers.Next;
        end;
      end
      else
      begin
         ShowMessage('The shop is not in this mall');
      end;
end;

end.
```

ANNEXURE G: SOLUTION FOR QUESTION 3

Object class:

```
unit GiftVoucher_U;
interface
uses SysUtils;
type

  TGiftVoucher = class(TObject)

  private
  var
    fVoucherNumber: integer;
    fName: String;
    fBalance: real;

  public
    constructor create(iVoucherNum: integer; sName: String; rBalance:
real);
    function getBalance: real;
    function getVoucherNumber: integer;
    function isSufficient(rAmount: real):boolean;
    procedure updateBalance(rAmount: real);
    function toString(): String;
  end;

implementation

{ TGiftVoucher }

// ========================================================================
// Question 3.1.1          4 marks
// ========================================================================
constructor TGiftVoucher.create(iVoucherNum: integer; sName: String;
  rBalance: real);
begin
  fVoucherNumber := iVoucherNum;
  fName := sName;
  fBalance := rBalance;
end;

// ========================================================================
// Question 3.1.2          2 marks
========================================================================
function TGiftVoucher.getVoucherNumber: integer;
begin
  Result := fVoucherNumber;
end;

// ========================================================================
// Question 3.1.2          2 marks
========================================================================
function TGiftVoucher.getBalance: real;
begin
  Result := fBalance;
end;
```



```
// ============================================================================
// Question 3.1.3          5 marks
// ============================================================================
function TGiftVoucher.isSufficient(rAmount: real): boolean;
begin
 if rAmount <= fBalance then
  begin
    Result := True;
  end
  else
  begin
    Result := False;
  end;
  // Alternative: Result := fBalance >= rAmount;
end;

// ============================================================================
// Question 3.1.4          3 marks
// ============================================================================
procedure TGiftVoucher.updateBalance(rAmount: real);
begin
    fBalance := fBalance - rAmount;
end;

// ============================================================================
// Question 3.1.5          4 marks
// ============================================================================
function TGiftVoucher.toString: String;
begin
  Result := 'Voucher number: ' + IntToStr(fVoucherNumber)
    + #13 + 'Customer name: ' + fName + #13 + 'Available balance: ' +
    FloatToStrF(fBalance, ffCurrency, 8, 2);
end;

end.
```


Main Form Unit:

```
unit Question3_U;

interface

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

type
  TfrmQuestion3 = class(TForm)
    pnlQ3Heading: TPanel;
    grbQ3_2_1: TGroupBox;
    grbQ3_2_2: TGroupBox;
    Label2: TLabel;
    cmbQ3_2_1: TComboBox;
    btnQ3_2_1: TButton;
    Label5: TLabel;
```

```
    edtQ3_2_2: TEdit;
    btnQ3_2_2_b: TButton;
    Label6: TLabel;
    edtQ3_2_1: TEdit;
    Image1: TImage;
    redQ3_2_1: TRichEdit;
    PageControl1: TPageControl;
    tshQ3_2_1: TTabSheet;
    tshQ3_2_2: TTabSheet;
    lblQ3_2_2: TLabel;
    pnlQ3_2_2: TPanel;
    Label7: TLabel;
    btnQ3_2_2_a: TButton;
    Panel1: TPanel;
    Label1: TLabel;
    edtQ3_VoucherNum: TEdit;
    procedure btnQ3_2_1Click(Sender: TObject);
    procedure btnQ3_2_2_bClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure btnQ3_2_2_aClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmQuestion3: TfrmQuestion3;
  objGiftVoucher: TGiftVoucher;

implementation

{$R *.dfm}
// ========================================================================
// Question 3.2.1          6 marks
// ========================================================================
procedure TfrmQuestion3.btnQ3_2_1Click(Sender: TObject);
var
  sName: String;
  rAmount: real;
  iVoucherNum: integer;
begin
  // Provided code
  redQ3_2_1.Clear;
  // Question 3.2.1
  iVoucherNum := StrToInt(edtQ3_VoucherNum.Text);
  sName := edtQ3_2_1.Text;
  rAmount := StrToFloat(cmbQ3_2_1.Text);
  objGiftVoucher := TGiftVoucher.create(iVoucherNum, sName, rAmount);
  redQ3_2_1.Lines.Add(objGiftVoucher.toString);
end;
```

```
// ============================================================================
// Question 3.2.2 (a)       5 marks
// ============================================================================
procedure TfrmQuestion3.btnQ3_2_2_1Click(Sender: TObject);
var
  iVNum: integer;
begin
  // Question 3.2.2
  iVNum := StrToInt(edtQ3_VoucherNum.Text);
  if iVNum = objGiftVoucher.getVoucherNumber then
  begin
    pnlQ3_2_2.Caption := FloatToStrF(objGiftVoucher.getBalance,
    ffCurrency, 8, 2);
    btnQ3_2_2_2.Enabled := true;
  end;
end;

// ============================================================================
// Question 3.2.2 (b)       9 marks
// ============================================================================
procedure TfrmQ3.btnQuestion3_2_2_bClick(Sender: TObject);
var
  rPurchaseAmount, rBalanceOnCard, rAmountOutstanding: real;
begin
  // Provided code
  edtQ3_2_2.SetFocus;

  // Question 3.2.1 - Enter your code here

  rPurchaseAmount:= StrToFloat(edtQ3_2_2.Text);
  if objGiftVoucher.isSufficient(rPurchaseAmount) = True then
  begin
    lblQ3_2_2.Caption := ('Gift voucher used successfully');
    objGiftVoucher.updateBalance(rPurchaseAmount);
  end
  else
  begin
  rAmountOutstanding := rPurchaseAmount - objGiftVoucher.getBalance;
     lblQ3_2_2.Caption := ('Amount owed by you to complete the purchase
           - ' + FloatToStrF(rAmountOutstanding, ffCurrency, 8, 2));
        objGiftVoucher.updateBalance(objGiftVoucher.getBalance);
  end;
  rBalanceOnCard := objGiftVoucher.getBalance;
  pnlQ3_2_2.Caption := FloatToStrF(rBalanceOnCard, ffCurrency, 8, 2);
end;

// Provided code
procedure TfrmQuestion3.FormCreate(Sender: TObject);
begin
  btnQ3_2_2_b.Enabled := false;
  PageControl1.TabIndex := 0;
end;

end.
```

ANNEXURE H: SOLUTION FOR QUESTION 4

```
unit Question4_U;

interface

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

type
  TfrmQuestion4 = class(TForm)
    btnQ4_1: TButton;
    redQ4: TRichEdit;
    rgpQ4: TRadioGroup;
    btnQ4_2: TButton;
    procedure FormActivate(Sender: TObject);
    procedure btnQ4_1Click(Sender: TObject);
    procedure btnQ4_2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmQuestion4: TfrmQuestion4;

  arrShopTypes: array [1 .. 8] of String = (
    'Electronics',
    'Jewellery',
    'Furniture',
    'Clothing',
    'Restaurant',
    'Toys',
    'Laundry',
    'Gifts'
  );
  arrShops: array [1 .. 10] of String;
  iLevel: integer;

implementation
{$R *.dfm}
// ========================================================================
// Question 4.1          15 marks
// ========================================================================
procedure TfrmQuestion4.btnQ4_1Click(Sender: TObject);
var
  iC, iR, iNumber: integer;
  tFile: TextFile;
  sLine: String;
begin
  redQ4.Clear;
  redQ4.Lines.Add('Level: ' + #9 + ' Shops:' + #9 +
                  'Number of shops per level:');
 redQ4.Lines.Add('================================================');
```



```
  AssignFile(tFile, rgpQ4.Items[rgpQ4.ItemIndex] + '.txt');
  Reset(tFile);

  iLevel := 0;
  while not eof(tFile) do
  begin
    inc(iLevel);
    Readln(tFile, sLine);
    arrShops[iLevel] := sLine;

    sLine := intToStr(iLevel) + '.' + sLine;

    iNumber := length(sLine);
    while length(sLine) < 25 do
      insert(' ', sLine, 3);

    redQ4.Lines.Add(sLine + #9 + intToStr(iNumber));
  end;
end;
//============================================================================
// Question 4.2          15 marks
// ============================================================================
procedure TfrmQuestion4.btnQ4_2Click(Sender: TObject);
const
  Shops = 'EJFCRTLG';
var
  iC, iR, iTotal, iIndex: integer;
  sShopType, sShop, sResults: String;

begin
  // Question 4.2
  redQ4.Lines.Add(' ');
  sShopType := uppercase(InputBox('Enter type of shop, e.g. F',
      'Valid shop types: C E F G J L R T', 'F'));
  iIndex := pos(sShopType, Shops);
  if iIndex = 0 then
    Showmessage('The valid shop types are : ' + #13 + 'C E F G J L R T')
  else
  begin
    redQ4.Lines.Add('Type of shop: ' + arrShopTypes[iIndex] + #13 +
                                    'Number per level');
    for iR := 1 to iLevel  do
    begin
      iTotal := 0;
      sResults := intToStr(iR) + '.  ';
      for iC := 1 to length(arrShops[iR]) do
      begin
        sShop := arrShops[iR][iC];
        if (sShop = sShopType) then
          inc(iTotal);
      end;
      sResults := sResults + #9 + intToStr(iTotal);
      redQ4.Lines.Add(sResults); // row total
    end; // go to next row
  end; // else
end;
```

```
// Provided code - do not change
procedure TfrmQuestion4.FormActivate(Sender: TObject);
begin
  redQ4.Paragraph.TabCount := 10;
  redQ4.Paragraph.Tab[0] := 20;
  redQ4.Paragraph.Tab[1] := 70;
  redQ4.Paragraph.Tab[2] := 130;
  redQ4.Paragraph.Tab[3] := 200;
  redQ4.Paragraph.Tab[4] := 260;
  redQ4.Paragraph.Tab[5] := 320;
  redQ4.Paragraph.Tab[6] := 370;
  redQ4.Paragraph.Tab[7] := 430;
  redQ4.Paragraph.Tab[8] := 500;
  redQ4.Paragraph.Tab[9] := 570;
end;

end.
```