basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

NOVEMBER 2022

MARKING GUIDELINES

MARKS: 150

These marking guidelines consist of 29 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 11) include the marking grid for each question.

- **Annexures E, F, G** and **H** (pages 12 to 29) contain examples of solutions for Question 1 to Question 4 in programming code.

- Copies of **Annexures A, B, C, D** and **the summary for the marks of the learner** (pages 3 to 11) 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 – Properties]** Set the panel colour to yellow ✓ Set the font style to italic ✓ Change caption to ”I love programming!” ✓	**3**	
1.2	**Button - [1.2 – Leave month]** Extract month name / month index from cmbQ1_2 ✓ Check IF (sMonth = 'June') ✓ OR (sMonth = 'July') OR (sMonth = 'August') ✓ then Display message 'Company closed, select another month.' ✓ Deselect month from cmbQ1_2 (Index -1) ✓ Else ✓ Display message 'Your leave in ' + sMonth + ' has been granted.' ✓ ALTERNATIVE IF conditions: • IF iMonth IN [5,6,7] then • IF (iMonth >= 5) AND (iMonth <= 7) then **NOTES:** If candidates adjust the index according to month numbers, the range must be from 6 to 8	**7**	

1.3	**Button - [1.3 - Calculate]** Extract A and B from edit boxes✓ converted to real ✓ `C := sqrt✓(power(A,5)✓ + pi✓ * sqr(B)✓);` Alternative to functions: • Sqrt or Power(Value, 0.5) • Power (do not accept hard-coding) • Pi or 22/7 or 3.14 • Sqr or Power(B, 2) or (B * B) Display Truncated (Floor or Trunc) ✓ value of C converted to String ✓ **NOTE:** Penalise once only for a logical mistake	**8**	
1.4	**Button - [1.4 – Marks]** Initialise counter and total ✓ Input mark ✓ Start conditional loop✓ to test mark <> -1 ✓ Increase counter variable ✓ Add mark to total mark ✓ Display the subject number ✓ and mark ✓ Input next mark ✓ Calculate average mark ✓ Display average mark in output area converted to String formatted to TWO decimal places ✓ **NOTES:** If repeat until used the condition will be: until mark = -1 If a single input is used inside the loop, an IF must be used inside the loop to prevent -1 to be accepted as a mark	**11**	

1.5	**Button - [1.5 – Anagram]** If length of two word strings are equal ✓ Loop ✓ from 1 to length of word 1 string ✓ Use character at index ✓ Find position ✓ of character in word 2 ✓ Delete character at position ✓ from word 2 ✓ If length of word 2 is null ✓ Display the message "The words form an anagram." ✓ Else Display a message "The words do not form an anagram." ✓ **Concepts**: Loop through length of first word //2 Extract character at index //1 Find position of character in second word //2 <u>Mechanism to match words</u>: //3 • Overwrite char at index word1 and char position in word2 with space • Test and change flag if Not found (flag set before loop) • Delete character from second word (if length tested same before the loop) <u>Test anagram passed and display messages</u>: //3 IF words are equal / word 2 = '' // flag still true ShowMessage IS anagram ShowMessage NOT anagram **Sorting as an alternative**: Sort characters in first word: Double looping //2 Comparing characters within the word //2 Swopping characters //3 Sorting characters in second word //1 <u>Test and display messages</u>: //3 IF words are equal ShowMessage IS anagram ShowMessage NOT anagram	**11**	
	TOTAL SECTION A:	**40**	

ANNEXURE B

QUESTION 2: MARKING GRID – SQL AND DATABASE PROGRAMMING

CENTRE NUMBER:	EXAMINATION NUMBER:		
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
2.1	**SQL statements**		

2.1.1	**Button [2.1.1 – Bloemfontein clients]**
	`SELECT * ✓`
	`FROM tblClients ✓`
	`WHERE City = "Bloemfontein" ✓`
	`ORDER BY ClientSurname ✓`
		4	

2.1.2	**Button [2.1.2 – First month]**
	`SELECT CollectionID, CollectionDate,`
	`NumberOfCans FROM tblCanCollection ✓`
	`WHERE Month ✓ (CollectionDate) = 1 ✓`
	
	`Alternative:`
	`Extracting month using String operators`
		3	

2.1.3	**Button [2.1.3 – Search]**
	`SELECT * FROM tblClients ✓`
	`WHERE ClientID LIKE ✓ "'+ sLetter ✓ +'%" ✓`
	
	`Alternative:`
	`WHERE left(ClientID, 1) = quotedStr(sLetter)`
		4	

2.1.4	**Button [2.1.4 – Clients' earnings]**
	`SELECT ClientName,`
	`FORMAT(SUM ✓ (NumberOfCans /76 * 8) ✓,`
	`"Currency") ✓`
	`AS ✓ [Total Amount] ✓`
	`FROM tblCanCollection A, tblClients B ✓`
	`WHERE A.ClientID = B.ClientID ✓`
	`GROUP BY ClientName ✓`
		8	

2.1.5	**Button [2.1.5 – Update]**
	`UPDATE tblCanCollection ✓`
	`SET NumberOfCans = 250 ✓, Paid = False ✓`
	`WHERE CollectionID = "C003" ✓`
		4	

	Subtotal:	**23**	

QUESTION 2: MARKING GRID (CONT.)

2.2	**DATABASE MANIPULATION using Delphi code**		
2.2.1	**Button [2.2.1 – Insert]** Insert/append mode ✓ Assign to correct field names ✓ Using the correct values for each field ✓ Post ✓ tblClients.Insert; tblClients['ClientID'] := 'CHA01'; tblClients['ClientName'] := 'Charles'; tblClients['ClientSurname'] := 'du Boit'; tblClients['Address'] := '24 Van Wouw Street'; tblClients['City'] := 'Cape Town'; tblClients.Post;	**4**	
2.2.2	**Button [2.2.2 – Percentage]** Display the name and surname of the client selected ✓ Initialise variables ✓ Go to the first record in the tblCanCollection table ✓ Use a loop to step through the tblCanCollection table ✓ Test year = year selected ✓ Increment company total of collections for the year ✓ if (ClientID in tblCanCollection = ClientID in tblClients) ✓ Increment client total of collections for the year ✓ Move to the next record in tblCanCollection ✓ *End loop (tblCanCollection)* Calculate percentage of client collection as part of the total collection of company ✓ (rPercentage := rTotalClient / rTotalCompany * 100) Display total client collection value formatted to String ✓ Display company collection value formatted to String ✓ Display percentage of client collection formatted to two decimals with % symbol added to result ✓	**13**	
	Subtotal:	**17**	
	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 Create** Constructor heading with three correct parameters ✓and correct data types for parameters ✓ Assign parameter values to the three attributes ✓✓ (fPlantCode, fNumberOfPanels, fPowerPerPanel) Set season attribute to "Summer" ✓	**5**	
3.1.2	**procedure incNumOfPanels** Procedure heading ✓ with integer parameter ✓ fNumberOfPanels := fNumberOfPanels ✓ + parameter ✓	**4**	
3.1.3	**procedure setSeason** procedure heading ✓ with string type parameter ✓ fSeason = parameter ✓	**3**	
3.1.4	**function calculateCapacity** function heading with real data type ✓ Test if fSeason = 'Summer' ✓ hours = 10 ✓ else if fSeason = 'Winter' ✓ hours = 6 ✓ else hours = 8 ✓ Result = ✓ fNumberOfPanels * fPowerPerPanel * hours ✓	**8**	
3.1.5	**function toString** Function declared with string as return data type ✓ All attributes part of string to return ✓ In correct format with text and with #13 ✓	**3**	
	Subtotal: Object class	**23**	

QUESTION 3: MARKING GRID (CONT.)

QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
3.2.1	**Button [3.2.1 – Instantiate object]** Extract the data from the components ✓ converted to correct data types ✓ objPlant := ✓ TSolarPowerPlant.create ✓ Use three arguments ✓ `(PlantCode,NumberOfPanels,PowerPerPanel)` Display information in the rich edit using toString method ✓	**6**	
3.2.2	**Button [3.2.2 – Increase panels]** Call incNumOfPanels method ✓ using value from the spin edit as an argument ✓ Display plant code using getPlantCode method ✓ and number of panels using getNumOfPanels method, converted to string ✓	**4**	
3.2.3	**Button [3.2.3 – Update season]** Call setSeason method ✓ using the season selected from the combo box as an argument ✓ Display information in the rich edit using toString method ✓	**3**	
3.2.4	**Button [3.2.4 – Calculate]** Display a suitable description ✓ including the season, using the getSeason method ✓ Display the result of the calculateCapacity method ✓, converted to string with "kW" added as unit ✓	**4**	
	Subtotal Form class:	**17**	
	TOTAL SECTION C:	**40**	

ANNEXURE D

QUESTION 4: MARKING GRID – PROBLEM-SOLVING

CENTRE NUMBER:	**EXAMINATION NUMBER:**

QUESTION	**DESCRIPTION**	**MAX MARKS**	**LEARNER'S MARKS**
4.1	**Button [4.1 – Display]** Outer loop `I` from 1 to length of array ✓ Or 10 Start output string with name[`I`] ✓ and add tab (#9) to output string Inner loop `J` from 1 to length of arrVending[`J`] ✓ OR 15 Join characters from arrVending to output string ✓ Display output string on rich edit ✓	**5**	
4.2	**Button [4.2 – Calculate]** Initialise Max variable to 0 ✓ Outer loop `I` from 1 to length of arrNames (10) ✓ Initialise amount paid to 0 ✓ Inner loop `J` from 1 to length of arrVending[`I`] (15) ✓ Test if item recycled is a bottle ✓ Increment amount paid with 2.15 ✓ Test if item is a can Increment amount paid with 0.75 ✓ *End inner loop* Display name and amount paid, converted to currency ✓ **Alternative 1**: Test if Amount >= Max ✓ if Amount > Max ✓ Set Max to amount ✓and clear output ✓ Add name and amount to Max output string ✓ *End outer loop* Display Max output string in redQ4 ✓ **Alternative 2 (using parallel array)**: Test if Amount > Max //1 Set Max to amount //1 Save learner amount in parallel array //1 *End outer loop* Loop from 1 to 10 //1 If Amount = Max //1 Display name and amount //1	**14**	

4.3	**Button [4.3 – Add item]** Test if name OR item are NOT selected ✓ Display message if name or item not selected ✓ Extract item from component ✓ Extract OR determine row(index) of name ✓ <u>PLACING ITEM</u>: Loop index from 1 to 15 ✓ Test if arrVending empty at index ✓ Place item in arrVending ✓ Flag / Break ✓ Display updated array ✓ <u>NO SPACE MESSAGE</u>: Check flag OR use another mechanism to determine availability of space ✓ Display message for no space available ✓	**11**	

	TOTAL SECTION D: **GRAND TOTAL:**	**30** **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                3 marks
// ================================================================
procedure TfrmQuestion1.btnQ1_1Click(Sender: TObject);
begin
  // Question 1.1

   pnlQ1_1.Color := clYellow;
   pnlQ1_1.Font.Style := [fsItalic];
   pnlQ1_1.Caption := 'I love programming!';
end;
//================================================================
// Question 1.2                7 marks
// ================================================================
procedure TfrmQuestion1.btnQ1_2Click(Sender: TObject);
begin
  // Question 1.2
  if (cmbQ1_2.ItemIndex > 4) AND (cmbQ1_2.ItemIndex < 8) then
    begin
      lblQ1_2.Caption :='Company closed, select another
                         month.';
      cmbQ1_2.ItemIndex := -1;
    end
  else
    lblQ1_2.Caption :='Your leave in ' + cmbQ1_2.text +
                                ' has been granted.';
end;

// ================================================================
// Question 1.3                 8 marks
// ================================================================
procedure TfrmQuestion1.btnQ1_3Click(Sender: TObject);
var
  rA,rB,rC : Real;
begin
  // Question 1.3
  rA := StrToFloat(edtQ1_3_1.Text);
  rB := StrToFloat(edtQ1_3_2.Text);
  rC := Sqrt(Power(rA,5)+ pi * Sqr(rB));

  edtQ1_3_3.Text := IntToStr(Trunc(rC));
end;

// ================================================================
// Question 1.4              11 marks
// ================================================================
procedure TfrmQuestion1.btnQ1_4Click(Sender: TObject);
var
   iMark, i, iTotal : Integer;
   rAverage : Real;
begin
  // Provided code
  redQ1_4.Clear;
```

```
  // Question 1.4
  iMark := StrToInt(InputBox('Learner marks','Enter mark for
                              subject : 1',''));
  i := 0;
  iTotal := 0;
  while iMark > -1 do
    begin
      Inc(i);
      iTotal := iTotal + iMark;
      redQ1_4.Lines.Add('Subject ' + IntToStr(i) + ' : ' +
                        IntToStr(iMark));
      iMark := StrToInt(InputBox('Learner marks','Enter mark for
                        subject : ' + IntToStr(i+1), ''));
    end;
  rAverage:= iTotal / i;
  redQ1_4.Lines.Add('Average mark:' +
                    FloatToStrF(rAverage,ffFixed,5,2));
end;

// ================================================================
// Question 1.5          11 marks
// ================================================================
procedure TfrmQuestion1.btnQ1_5Click(Sender: TObject);
var
  sWord1,sWord2 : String;
  I : integer;
begin
  //Provided code
  redQ1_5.Clear;
  sWord1 := Lowercase(edtQ1_5_1.Text);
  sWord2 := Lowercase(edtQ1_5_2.Text);

  // Question 1.5

  if length(sWord1) = length(sWord2) then
  begin
    for i := 1 to length(sWord1) do
      delete(sWord2,(pos(sWord1[i],sWord2)),1);
    if sWord2 = '' then
      memQ1_5.Lines.add( 'The words form an anagram.')
    else
      memQ1_5.Lines.add('The words do not form an anagram.');
  end
  else
    memQ1_5.Lines.add('The words do not form an anagram.');
end;

end.
```

ANNEXURE F: SOLUTION FOR QUESTION 2

```
unit Question2_U;

interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
  Controls, Forms, Dialogs, DB, ExtCtrls, StdCtrls,
  ComCtrls, Grids, DBGrids, Buttons, ADODB, Math, DateUtils,
  ConnectDB_U;

type
  TfrmQuestion2 = class(TForm)
    pgcDBAdmin: TPageControl;
    tabsQ2SQL: TTabSheet;
    btnQ2_1_5: TBitBtn;
    grpresults: TGroupBox;
    btnQ2_1_1: TBitBtn;
    btnQ2_1_2: TBitBtn;
    b: TTabSheet;
    grpQ2_2_1: TGroupBox;
    redQ2_2_2: TRichEdit;
    btnQ2_2_2: TButton;
    dbgClients: TDBGrid;
    dbgCollections: TDBGrid;
    grpQ2_2_2: TGroupBox;
    btnQ2_2_1: TButton;
    btnQ2_1_4: TBitBtn;
    pnlBtns: TPanel;
    bmbClose: TBitBtn;
    bmbRestoreDB: TBitBtn;
    dbgrdSQL: TDBGrid;
    btnQ2_1_3: TButton;
    rgpQ2_2_2: TRadioGroup;
    cmbQ2_2_2: TComboBox;
    lblQ2_2_2: TLabel;
    procedure btnQ2_1_1Click(Sender: TObject);
    procedure btnQ2_1_2Click(Sender: TObject);
    procedure btnQ2_1_4Click(Sender: TObject);
    procedure btnQ2_1_3Click(Sender: TObject);
    procedure btnQ2_1_5Click(Sender: TObject);
    procedure btnQ2_2_2Click(Sender: TObject);
    procedure btnQ2_2_1Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure bmbRestoreDBClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmQuestion2: TfrmQuestion2;
  dbCONN : TConnection;
```

```
// --- Global variables provided ---
  tblClients, tblCanCollection : TADOTable;

implementation

{$R *.dfm}

procedure TfrmQuestion2.bmbRestoreDBClick(Sender: TObject);
begin
// Restores the Database
  dbCONN.RestoreDatabase;
  redQ2_2_2.Clear;
  dbCONN.SetupGrids(dbgClients, dbgCollections, dbgrdSQL);
end;

//=============================================================
// Question 2.1.1          4 marks
//=============================================================

procedure TfrmQuestion2.btnQ2_1_1Click(Sender: TObject);
var
  sSQL1: String;
begin
// Question 2.1.1

  sSQL1 := 'SELECT * FROM tblClients ' +
           'WHERE City = "Bloemfontein"' +
           'ORDER BY ClientSurname';

// Provided code - do not change
  dbCONN.runSQL(sSQL1);
end;

//=============================================================
//Question 2.1.2          3 marks
//=============================================================

procedure TfrmQuestion2.btnQ2_1_2Click(Sender: TObject);
var
  sSQL2: String;
begin
// Question 2.1.2

  sSQL2 := 'SELECT CollectionID, CollectionDate, NumberOfCans ' +
           'FROM tblCanCollection ' +
           'WHERE Month(CollectionDate) = 1';

// Provided code - do not change
  dbCONN.runSQL(sSQL2);
end;
```

```
//=====================================================================
// Question 2.1.3          4 marks
//=====================================================================

procedure TfrmQuestion2.btnQ2_1_3Click(Sender: TObject);
var
  sSQL3, sLetter : String;
begin
// Question 2.1.3
// Provided code
  sLetter := InputBox('','Enter first letter of ClientID','');

    sSQL3 := 'SELECT * FROM tblClients ' + ✓
             'WHERE ClientID LIKE ✓ "'+ sLetter ✓ +'%"'; ✓

    // Alternative:
      // WHERE left(clientID, 1) = quotedStr(sLetter)

// Provided code - do not change
  dbCONN.runSQL(sSQL3);
end;

//=====================================================================
// Question 2.1.4          8 marks
//=====================================================================

procedure TfrmQuestion2.btnQ2_1_4Click(Sender: TObject);
var
  sSQL4: String;
begin
// Question 2.1.4

  sSql4 := 'SELECT ClientName, ' +
           'FORMAT(SUM(NumberOfCans / 76 * 8),"Currency") ' +
           'AS [Total Amount] ' +
           'FROM tblCanCollection A, tblClients B ' +
           'WHERE A.ClientID = B.ClientID ' +
           'GROUP BY ClientName';

// Provided code - do not change
  dbCONN.runSQL(sSQL4);
end;

//=====================================================================
// Question 2.1.5          4 marks
//=====================================================================

procedure TfrmQuestion2.btnQ2_1_5Click(Sender: TObject);
var
  sSql5 : String;
  bChange : boolean;
begin
// Question 2.1.5
```

```
  sSql5 := 'UPDATE tblCanCollection ' +
           'SET NumberOfCans = 250, Paid = False ' +
           'WHERE CollectionID = "C003"';

// Provided code - do not change
dbCONN.ExecuteSQL(sSQL5, bChange);

if bChange then
 begin
   MessageDlg('Database has been updated.', mtInformation, [mbOK], 0);
 end;
end;

//=====================================================================
// Question 2.2.1          4 marks
//=====================================================================

procedure TfrmQuestion2.btnQ2_2_1Click(Sender: TObject);
begin
// Question 2.2.1

tblClients.Insert;
tblClients['ClientID'] := 'CHA01';
tblClients['ClientName'] := 'Charles';
tblClients['ClientSurname'] := 'du Boit';
tblClients['Address'] := '24 Van Wouw Street';
tblClients['City'] := 'Cape Town';
tblClients.Post;
end;

//=====================================================================
// Question 2.2.2          13 marks
//=====================================================================

procedure TfrmQuestion2.btnQ2_2_2Click(Sender: TObject);
var
  iCompanyTotal, iClientTotal : Integer;
  sYear : String;
  rPercentage : Real;
begin
// Provided code - do not change
  redQ2_2_2.Clear;
  sYear := rgpQ2_2_2.Items[rgpQ2_2_2.ItemIndex];
// ============================================
// Question 2.2.2

  redQ2_2_2.Lines.Add(tblClients['ClientName']+'
                  '+tblClients['ClientSurname']+#13);
  iCompanyTotal := 0;
  iClientTotal := 0;
  tblCanCollection.First;
  while NOT (tblCanCollection.Eof) do
    begin
      if sYear = IntToStr(YearOf(tblCanCollection['CollectionDate'])) then
        begin
```

```
          iCompanyTotal := iCompanyTotal +
                           tblCanCollection['NumberOfCans'];

        if (tblCanCollection['ClientID'] = tblClients['ClientID']) then
            iClientTotal := iClientTotal +
                            tblCanCollection['NumberOfCans'];
         end;
       tblCanCollection.Next;
     end;
  rPercentage := iClientTotal / iCompanyTotal * 100;
  redQ2_2_2.Lines.Add('Client collection for ' + sYear + ': ' + #9 +
                      IntToStr(iClientTotal));
  redQ2_2_2.Lines.Add('Company collection for ' + sYear + ': ' + #9 +
                      IntToStr(iCompanyTotal));
  redQ2_2_2.Lines.Add('Percentage collected by client: ' + #9 +
                FloatToStrF(rPercentage, ffFixed, 3, 2) + '%');
end;

procedure TfrmQuestion2.FormCreate(Sender: TObject);
begin
  redQ2_2_2.Paragraph.TabCount := 2;
  redQ2_2_2.Paragraph.Tab[0] := 180;
  redQ2_2_2.Paragraph.Tab[1] := 150;
end;

procedure TfrmQuestion2.FormShow(Sender: TObject);
begin
  // Sets up the connection to database and opens the tables.
  dbCONN := TConnection.Create;
  dbCONN.dbConnect;
  tblClients := dbCONN.tblOne;
  tblCanCollection := dbCONN.tblMany;
  dbCONN.setupGrids(dbgClients, dbgCollections, dbgrdSQL);
  pgcDBAdmin.ActivePageIndex := 0;
end;

end.
```

ANNEXURE F: SOLUTION FOR QUESTION 3

Object class:

```
unit SolarPowerPlant_U;

interface

type
  TSolarPowerPlant = class(TObject)
  private
  var
    fPlantCode: String;
    fNumberOfPanels: Integer;
    fPowerPerPanel: Real;
    fSeason: String;
  public
    // Provide code
    function getPlantCode : String;
    function getNumOfPanels : Integer;
    function getSeason : String;
    // Code here

    constructor create(sPlantCode: String; iNumOfPanels: Integer;
                       rPowerPerPanel:Real);
    procedure incNumOfPanels(iNumber : Integer);
    procedure setSeason(sNewSeason : String);
    function calculateCapacity : Real;
    function toString: String;
  end;

implementation

{ TStorage }

uses
  SysUtils, Math;

{ TSolarPowerPlant }

// ========================================================
// Question 3.1.1          5 marks
// ========================================================

constructor TSolarPowerPlant.create(sPlantCode: String;
  iNumOfPanels: Integer; rPowerPerPanel: Real);
begin
  fPlantCode := sPlantCode;
  fNumberOfPanels := iNumOfPanels;
  fPowerPerPanel := rPowerPerPanel;
  fSeason := 'Summer';
end;
```

```
// ========================================================
// Question 3.1.2            4 marks
// ========================================================

procedure TSolarPowerPlant.incNumOfPanels(iNumber: integer);
begin
  fNumberOfPanels := fNumberOfPanels + iNumber;
end;

// ========================================================
// Question 3.1.3            3 marks
// ========================================================

procedure TSolarPowerPlant.setSeason(sNewSeason: String);
begin
  fSeason := sNewSeason;
end;

// ========================================================
// Question 3.1.4            8 marks
// ========================================================

function TSolarPowerPlant.calculateCapacity: Real;
var
  iHours : Integer;
begin

  if fSeason = 'Summer' then
  begin
    iHours := 10;
  end
  else if fSeason = 'Winter' then
  begin
    iHours := 6;
  end
  else
  begin
    iHours := 8;
  end;

  result := (fNumberOfPanels * fPowerPerPanel) * iHours;

end;
```

```
// ========================================================
// Question 3.1.5            3 marks
// ========================================================

function TSolarPowerPlant.toString: String;
var
  sString : String;
begin
  sString := 'Plant code: ' + fPlantCode + #13;
  sString := sString + 'Number of panels: ' + intToStr(fNumberOfPanels) +
#13;
  sString := sString + 'Power per panel: ' + floatToStr(fPowerPerPanel) +
#13;
  sString := sString + 'Season: ' + fSeason + #13;
  result := sString;
end;
// ========================================================
// Provided code
// ========================================================

function TSolarPowerPlant.getPlantCode: String;
begin
  result := fPlantCode;
end;

function TSolarPowerPlant.getNumOfPanels: Integer;
begin
  result := fNumberOfPanels;
end;

function TSolarPowerPlant.getSeason: String;
begin
  result := fSeason;
end;

end.
```

Main form unit:

```
unit Question3_U;

interface

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

type
  TfrmQuestion3 = class(TForm)
    gbxQ3_2_1: TGroupBox;
    gbxQ3_2_3: TGroupBox;
    btnQ3_2_1: TButton;
    btnReset: TButton;
    gbxQ3_2_2: TGroupBox;
    btnQ3_2_2: TButton;
    edtQ3_2_1_Power: TEdit;
    Label2: TLabel;
    Label3: TLabel;
    sedQ3_2_2: TSpinEdit;
    Panel1: TPanel;
    Panel2: TPanel;
    btnQ3_2_3: TButton;
    Image1: TImage;
    Label6: TLabel;
    edtQ3_2_1_Code: TEdit;
    sedQ3_2_1: TSpinEdit;
    Label4: TLabel;
    cmbQ3_2_3: TComboBox;
    Label5: TLabel;
    gbxQ3_2_4: TGroupBox;
    btnQ3_2_4: TButton;
    redQ3: TRichEdit;
    procedure btnQ3_2_1Click(Sender: TObject);
    procedure btnResetClick(Sender: TObject);
    procedure btnQ3_2_2Click(Sender: TObject);
    procedure btnQ3_2_3Click(Sender: TObject);
    procedure btnQ3_2_4Click(Sender: TObject);
  private
  public
  end;
var
  frmQuestion3: TfrmQuestion3;

implementation
{$R *.dfm}

uses
  SolarPowerPlant_U;

var
  objPlant: TSolarPowerPlant;
```

```
// ========================================================
// Question 3.2.1            6 marks
// ========================================================

procedure TfrmQuestion3.btnQ3_2_1Click(Sender: TObject);
begin
  // Provided code
  redQ3.Clear;
  // Question 3.2.1

  objPlant:= tSolarPowerPlant.create(edtQ3_2_1_Code.Text,
     sedQ3_2_1.Value, strToFloat(edtQ3_2_1_Power.Text));
  redQ3.Lines.Add(objPlant.toString);
end;

// ========================================================
// Question 3.2.2            4 marks
// ========================================================

procedure TfrmQuestion3.btnQ3_2_2Click(Sender: TObject);
var
  rUpdatedPower: Real;
begin
  // Provided code
  redQ3.Clear;
  // Question 3.2.2

  objPlant.incNumOfPanels(sedQ3_2_2.Value);
  redQ3.Lines.Add('Plant code: ' + objPlant.getPlantCode);
  redQ3.Lines.Add('Number of panels: ' +
                IntToStr(objPlant.getNumOfPanels));

end;
// ========================================================
// Question 3.2.3            3 marks
// ========================================================

procedure TfrmQuestion3.btnQ3_2_3Click(Sender: TObject);
begin
  // Provided code
  redQ3.Clear;
  // Question 3.2.3

  objPlant.setSeason(cmbQ3_2_3.Text);
  redQ3.Lines.Add(objPlant.toString)

end;
```

```
// ========================================================
// Question 3.2.4            4 marks
// ========================================================

procedure TfrmQuestion3.btnQ3_2_4Click(Sender: TObject);
begin
  // Provided code
  redQ3.Clear;
  // Question 3.2.4

  redQ3.Lines.Add('The maximum generation capacity per day in ' +
                  objPlant.getSeason + ': ');
  redQ3.Lines.Add(floatToStr(objPlant.calculateCapacity) + ' kW');

end;

// ========================================================
// Provided code
// ========================================================

procedure TfrmQuestion3.btnResetClick(Sender: TObject);
begin
  objPlant.Free;
  edtQ3_2_1_Power.Clear;
  edtQ3_2_1_Code.Clear;
  sedQ3_2_1.Value := 15;
  sedQ3_2_2.Value := 50;
  redQ3.Clear;
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, jpeg;

type
  TfrmQuestion4 = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    btnQ4_3: TButton;
    redQ4: TRichEdit;
    btnQ4_1: TButton;
    GroupBox1: TGroupBox;
    rgpQ4: TRadioGroup;
    btnQ4_2: TButton;
    Image1: TImage;
    lstQ4: TListBox;
    GroupBox2: TGroupBox;
    GroupBox3: TGroupBox;
    procedure btnQ4_3Click(Sender: TObject);
    procedure btnQ4_1Click(Sender: TObject);
    procedure btnQ4_2Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
  private
    { Private declarations }

  public
    { Public declarations }
  end;

var
  frmQuestion4: TfrmQuestion4;

  arrNames: array [1 .. 10] of String = (
    'Ruth',
    'Nicole',
    'Loyiso',
    'Chris',
    'William',
    'Thabo',
    'Vusi',
    'Peter',
    'Jenny',
    'Tommy'
  );
```

```
  arrVending: array [1 .. 10, 1 .. 15] of String =
 (('C', '', '', '', '', '', '', '', '', '', '', '', '', '', ''),
    ('B', 'B', 'B', 'C', 'C', 'C', 'B', 'B', 'B', 'C', 'C', 'C', 'C', 'C',
''),
    ('', '', '', '', '', '', '', '', '', '', '', '', '', '', ''),
    ('C', 'C', '', '', '', '', '', '', '', '', '', '', '', '', ''),
    ('B', 'B', 'C', 'C', 'B', 'B', 'C', 'C', 'C', 'C', 'B', 'C', 'C', 'B',
''),
    ('C', 'C', 'B', '', '', '', '', '', '', '', '', '', '', '', ''),
    ('C', 'B', '', '', '', '', '', '', '', '', '', '', '', '', ''),
    ('B', 'B', '', '', '', '', '', '', '', '', '', '', '', '', ''),
    ('C', '', '', '', '', '', '', '', '', '', '', '', '', '', ''),
    ('B', 'C', '', '', '', '', '', '', '', '', '', '', '', '', ''));
implementation

{$R *.dfm}

// ============================================================
// Question 4.1              5 marks
// ============================================================

procedure TfrmQuestion4.btnQ4_1Click(Sender: TObject);
var
  I, J: Integer;
  sLine: String;
begin
  // Provided code
  redQ4.Clear;
  redQ4.Lines.Add('-----------------------------');
  redQ4.Lines.Add('Names'+#9+'Items recycled');
  redQ4.Lines.Add('-----------------------------');

  // Question 4.1

  for I := 1 to length(arrNames) do
  begin
    sLine := arrNames[I] + #9;
    for J := 1 to length(arrVending[J]) do
    begin
      sLine := sLine + arrVending[I, J];
    end;
    redQ4.Lines.Add(sLine);
  end;

end;

// ============================================================
// Question 4.2              14 marks
// ============================================================

procedure TfrmQuestion4.btnQ4_2Click(Sender: TObject);
var
  I, J: Integer;
  rAmount, rMax: Real;
  iMax: Integer;
  arrTotal: Array [1 .. 10] of Real;
  sMaxs: String;
```

```
begin
  // Provided code
  redQ4.Clear;
  redQ4.Lines.Add('----------------------------');
  redQ4.Lines.Add('Names'+#9+'Total amount paid');
  redQ4.Lines.Add('----------------------------');

   // Question 4.2
   // Alternative
  {rMax := -1;
  for I := 1 to length(arrNames) do
  begin
    rAmount := 0;
    for J := 1 to length(arrVending[I]) do
      if arrVending[I, J] = 'B' then
       rAmount := rAmount + 2.15
       else   if arrVending[I, J] = 'C' then
       rAmount := rAmount + 0.75;

    if rAmount > rMax then
      rMax := rAmount;

   arrTotal[I] := rAmount;
   redQ4.Lines.Add(arrNames[I] + #9 + format('%8.2m', [rAmount]));
  end;//I

  //Provided code
  redQ4.Lines.Add('----------------------------');
  redQ4.Lines.Add('Highest payout(s):');
  redQ4.Lines.Add('----------------------------');

  //Code here

  for I := 1 to length(arrTotal) do
  begin
  redQ4.SelAttributes.Style:= [fsBold];
  redQ4.SelAttributes.Color:= clRed;
    if arrTotal[I] = rMax then
    begin
      redQ4.Lines.Add(arrNames[I] + #9 + format('%8.2m', [rMax]));
    end;

  end;}

rMax := -1;
for I := 1 to length(arrNames) do
  begin
    rAmount := 0;
    for J := 1 to length(arrVending[I]) do
      if arrVending[I, J] = 'B' then
         rAmount := rAmount + 2.15
        else if arrVending[I, J] = 'C' then
                    rAmount := rAmount + 0.75;

    redQ4.Lines.Add(arrNames[I] + #9 + format('%8.2m', [rAmount]));
```

```
    if rAmount >= rMax then
      begin
        if rAmount > rMax then
          begin
            rMax := rAmount;
            sMaxs := '';
          end;
        sMaxs := sMaxs + arrNames[I] + #9 + format('%8.2m',
                                                   [rAmount]) + #13;
      end;
  end;//I

  //Provided code
  redQ4.Lines.Add('---------------------------');
  redQ4.Lines.Add('Highest payout(s):');
  redQ4.Lines.Add('---------------------------');
  redQ4.Lines.Add(sMaxs);
end;

// Provided code
procedure TfrmQuestion4.FormShow(Sender: TObject);
begin
  redQ4.Paragraph.TabCount := 1;
  redQ4.Paragraph.Tab[0] := 70;
end;

// ============================================================
// Question 4.3                11 marks
// ============================================================

procedure TfrmQuestion4.btnQ4_3Click(Sender: TObject);
var
  iRow, iCol: Integer;
  sItem: String;
  bAdded: Boolean;
begin
  // Question 4.3
  // Alternative
 {bAdded := False;
  if lstQ4.ItemIndex = -1 then
  begin
    showMessage('Please select a name');
    exit;
  end
  else
  begin
      case rgpQ4.ItemIndex of
    -1:
      showMessage('Please select an item');
    else
     sItem := rgpQ4.Items[rgpQ4.ItemIndex][1];
   end;
```

```
      iRow := lstQ4.ItemIndex+1;
      iCol:=0;
      while (bAdded=false) AND ( iCol<15) do
      begin
       Inc(iCol);
           if arrVending[iRow,iCol] ='' then
           begin
             arrVending[iRow,iCol] := sItem;
             bAdded := True;
           end;
      end;
      if bAdded = false then
      begin
        showMessage('Machine is full.');
      end;

  end; }

  bAdded := False;
  if (lstQ4.ItemIndex > -1) AND (rgpQ4.ItemIndex > -1) then
  begin
    sItem := rgpQ4.Items[rgpQ4.ItemIndex][1];
    iRow := lstQ4.ItemIndex+1;
    iCol:=0;
    while (bAdded=false) AND ( iCol<15) do
     begin
       Inc(iCol);
       if arrVending[iRow,iCol] = '' then
          begin
            arrVending[iRow,iCol] := sItem;
            bAdded := True;
          end;
     btnQ4_1.Click;
     end;

     if bAdded = false then
     begin
       showMessage('Machine is full.');
     end;

  end
  else
    ShowMessage('Please select both a name and an item.');

  //Provided code
  rgpQ4.ItemIndex := -1;
  lstQ4.ItemIndex := -1;

end;

end.
```