

Need an amazing tutor?

www.teachme2.com/matric



Collected and collated by

teachme2

+



basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

**SENIOR CERTIFICATE/
NATIONAL SENIOR CERTIFICATE**

GRADE 12

INFORMATION TECHNOLOGY P1

NOVEMBER 2020

MARKING GUIDELINES

MARKS: 150

These marking guidelines consist of 28 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 3-10) include the marking grid for each question for using either one of the two programming languages.
- **Annexures E, F, G and H** (pages 12-28) contain examples of solutions for Questions 1 to 4 in programming code.
- Copies of **Annexures A, B, C and D** (pages 3-11) 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 – Colours] Change the font size of cmbQ1_1 to 12pt✓ Add blue to cmbQ1_1✓✓ Set item index to 0 / set text to Green✓	4	
1.2	Button [1.2 – Kite] Create variable for area (rArea)✓ Test if iDiagA > iDiagB✓ rArea = iDiagA * iDiagB ✓ / 2✓ Display rArea on label ✓ formatted to one decimal✓ Else✓ Display error message✓	8	
1.3	Button [1.3 – Binary number] Extract decimal number from edit box✓, convert to integer✓ Initialise output string for binary value✓ Loop while ✓ decimal number > 0✓ Remainder = decimal number MOD 2 ✓ (result 0 or 1) Decimal number ✓ = decimal number DIV 2 ✓ Add the remainder converted to a string✓ to beginning of output string ✓ Display binary number on label lblQ1_3 ✓ Alternatives: Loop while decimal number div 2 > 0 Repeat until decimal number = 0	11	
1.4	Button [1.4 – Word game] Initialise total to 0 ✓ Extract word from edit box ✓ and change to uppercase✓ Test if ✓ it is not a single word (has spaces) ✓ Clear edit box✓ Set focus to the edit box✓ Show error message✓ Else✓ Loop through word using length of word ✓ Extract character at loop variable✓ Test if it is a vowel ✓ add 3 to total ✓ Else Test for other alphabetical character✓ Add 2 to total ✓ Else Add 1 to total ✓ Display total value of word in richedit✓	17	
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 employees]	4	
	<pre>SELECT * FROM tblEmployees ORDER BY JobTitle, HourlyWage DESC</pre> <p>Concepts: SELECT * ✓ FROM tblEmployees ✓ ORDER BY JobTitle ✓ ,HourlyWage DESC ✓</p>		
2.1.2	Button [2.1.2 – Engineers]	4	
	<pre>SELECT EmployeeID, LastName, FirstName FROM tblEmployees WHERE JobTitle LIKE "%Engineer%"</pre> <p>Concepts: SELECT EmployeeID, LastName, FirstName ✓ FROM tblEmployees ✓ WHERE JobTitle LIKE ✓ "%Engineer%" ✓</p>		
2.1.3	Button [2.1.3 – Job titles]	3	
	<pre>SELECT DISTINCT JobTitle FROM tblEmployees OR SELECT JobTitle FROM tblEmployees GROUP BY JobTitle</pre> <p>Concepts: SELECT DISTINCT ✓ JobTitle ✓ FROM tblEmployees ✓ OR SELECT JobTitle FROM tblEmployees GROUP BY JobTitle</p>		
2.1.4	Button [2.1.4 – Remove records]	2	
	<pre>DELETE FROM tblHourLogs WHERE HoursWorked = 99</pre> <p>Concepts: DELETE FROM tblHourLogs ✓ WHERE HoursWorked = 99 ✓</p>		

QUESTION 2: MARKING GRID – CONTINUE

2.1.5	Button [2.1.5 – Overtime]	9	
	<pre>SELECT LastName, FORMAT(SUM((HoursWorked - 8) * HourlyWage * 2), "CURRENCY") AS OvertimeAmt FROM tblEmployees E, tblHourLogs H WHERE E.EmployeeID = H.EmployeeID AND HoursWorked > 8 GROUP BY LastName</pre> <pre>SELECT LastName, ✓ FORMAT(SUM ✓((HoursWorked - 8) ✓ * HourlyWage * 2 ✓), "CURRENCY") ✓ AS OvertimeAmt ✓ FROM tblEmployees E, tblHourLogs H ✓ WHERE E.EmployeeID = H.EmployeeID ✓ AND HoursWorked > 8 GROUP BY LastName ✓</pre> <p>Concepts: Retrieve LastName (1) Format to overtime pay as currency (1) Use SUM function correctly (1) Calculate number of hours worked overtime (1) Multiply result of above step by HourlyWage * 2 (1) Name calculated field OverTimeAmt (1) Extract data from tblEmployees and tblHourLogs (1) Test if there is a relationship between PK and FK (1) Group results by LastName (1)</p>		
	Subtotal:	22	

2.2	DATABASE MANIPULATION using Delphi code		
2.2.1	Button [2.2.1 – Employees with first aid] Display column headings // Provided Initialise counter to 0 Set tblEmployees to start reading first record ✓ Loop while not tblEmployees.Eof ✓ Test if tblEmployees['FirstAidTraining'] = True ✓ Display EmployeeID, LastName and JobTitle ✓ Increment counter by 1 ✓ Go to next record in tblEmployees ✓ Display counter ✓ converted to string	7	
2.2.2	Button [2.2.2 – Add new employee] tblEmployees.Insert ✓ tblEmployees['EmployeeID'] := 'EMP986'; ✓ tblEmployees['FirstName'] := 'Robert' ✓ tblEmployees['LastName'] := 'Laubscher' ✓ tblEmployees['HourlyWage'] := 195.00 ✓ tblEmployees['JobTitle'] := 'Marine Engineer' ✓ tblEmployees['FirstAidTraining'] := True ✓ tblEmployees.Post ✓ Alternatives tblEmployees['HourlyWage'] := 'R195.00'; tblEmployees['HourlyWage'] := FloatToStrF(195.00, ffCurrency, 6, 2); Concepts: Insert /Append (1) String variables – ID (1), FirstName and LastName (1) JobTitle and HourlyWage (1) FirstAidTraining (1) Post (1)	6	
2.2.3	Button [2.2.3 – Update hours worked] Get number of hours (iHours or sHours) from edtQ2_2_3✓, typecast to Integer tblHourLogs.Edit ✓ tblHourLogs['HoursWorked'] ✓ := iHours ✓ tblHourLogs.Post ✓	5	
	Subtotal:	18	
	TOTAL SECTION B:	40	

ANNEXURE C**QUESTION 3: MARKING GRID - OBJECT-ORIENTED PROGRAMMING**

QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
3.1.1	Constructor method: Header with correct parameter values (correct order, data types) ✓ Assign customer ID parameter value to fCustomerID ✓ Assign container size parameter value to fContainerSize ✓ Assign storage period parameter value to fStoragePeriod Set fAmountPaid to 0 ✓	4	
3.1.2	getAmountPaid method: Function heading with real/double as return data type ✓ fAmountPaid assigned to result ✓	2	
3.1.3	updateAmountPaid method: Procedure heading with real/double parameter value ✓ Increment fAmountPaid ✓ Using the parameter value ✓	3	
3.1.4	calculateCost method: Function declared with double/real return data type If containerSize = 'S' Set initial cost to 1000.00 Else if containerSize = 'M' Set initial cost to 1750.00 Else Set initial cost to 2500.00 Cost = storagePeriod * initial cost Percentage discount = 10% for each increment of 6 months using the floor/trunk/div function or other applicable code Test if discount > 50 Set discount = 50 Calculate discount value (Percentage x Amount) Return cost as result	11	

	Adapted marking approach <i>Allocate marks for the following concepts:</i> Function declared✓ with a return data type ✓ A test for the container size✓✓ Assigning the correct cost✓ per size✓ Calculate cost ✓ using the correct formula Determine percentage discount using months✓ Limit discount percentage to 50✓ Calculate discount value ✓ Return cost as result ✓		
3.1.5	toString method: Labels (Customer ID, Container size, Storage period, Amount paid) ✓ Correct attributes✓ Correct conversions (amountPaid – float; storagePeriod – integer) ✓	3	
	Subtotal: Object class	23	

ANNEXURE D**QUESTION 4: MARKING GRID–PROBLEM SOLVING**

CENTRE NUMBER:		EXAMINATION NUMBER:	
SECTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
4.1.1	Button [4.1.1 – Create harbour containers] Loop 50 times ✓ Generate real numbers ✓between 1 and 99 (inclusive)✓ Rounded to 1 decimal Store the values in the array arrContainers✓	4	
4.1.2	Button [4.1.2 – Display harbour containers] Set index to 0✓ Loop 5 times (rows) ✓ sLine := '' ✓ Loop 10 times (columns) ✓ Increment the index with 1 ✓ sLine := sLine + FloatToStr(arrContainers[iIndex]) ✓ + #9;✓ Display sLine in the rich edit redQ4_1_2✓ Alternative: sLine := sLine + FloatToStr(arrContainers[c+(10*(r-1))] (3) + #9; (1) Concept: Using loop variables (1) Calculation (1) Floattostr (1) Space (1)	8	

4.2	Button [4.2 – Containers loaded to be shipped] Set TotalTons to zero (0) ✓ Initialise string variable used for display ✓ Set array index (1) ✓ Loop index < 50 ✓ If (totalTons + weight of the containers at Index) ✓ <= 200 ✓ then Add weight of the containers at Index ✓ in array to TotalTons ✓ Add the weight of the containers at Index ✓ in array and add #9 ✓ to sLine ✓ Increment the index ✓ Outside loop: Display the weight of the containers loaded onto the ship in the richedit ✓ Display the total weight of the containers loaded on the panel ✓ Assign internal file name to external file name ✓ Rewrite ✓ Write the weight of the containers loaded onto the ship to the text file ✓ Close the text file ✓ Alternative For loop from 1 to 50 (3 marks)	18	
-----	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----	--

TOTAL SECTION D:	30	
GRAND TOTAL:	150	

SUMMARY OF LEARNER'S MARKS:

CENTER NUMBER:			LEARNER'S EXAM 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

```

unit Question1_U;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, ExtCtrls, Spin, pngimage, ComCtrls, Math,
  Buttons;

TfrmQuestion1 = class(TForm)
  GroupBoxQ1_1: TGroupBox;
  GroupBoxQ1_4: TGroupBox;
  edtQ1_4: TEdit;
  Label9: TLabel;
  redQ1_4: TRichEdit;
  GroupBoxQ1_3: TGroupBox;
  edtQ1_3: TEdit;
  btnQ1_3: TButton;
  Label10: TLabel;
  lblQ1_3: TLabel;
  btnQ1_1: TButton;
  cmbQ1_1: TComboBox;
  btnClose: TBitBtn;
  GroupBoxQ1_2: TGroupBox;
  lblQ1_2: TLabel;
  btnQ1_2: TButton;
  btnQ1_4: TButton;
  Image2: TImage;
  procedure btnQ1_2Click(Sender: TObject);
  procedure btnQ1_4Click(Sender: TObject);
  procedure btnQ1_3Click(Sender: TObject);
  procedure btnQ1_1Click(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;
var
  frmQuestion1: TfrmQuestion1;

implementation

{$R *.dfm}

//=====
//                               Question 1.1 - 4 marks
//=====
procedure TfrmQuestion1.btnQ1_1Click(Sender: TObject);
begin
  // Question 1.1
  cmbQ1_1.Font.Size := 12;
  cmbQ1_1.Items.Add('Blue');
  cmbQ1_1.ItemIndex := 0;
end;

```

```
//=====
//
//                                     Question 1.2 - 8 marks
//=====
procedure TfrmQuestion1.btnQ1_2Click(Sender: TObject);
var
    iDiagA, iDiagB: integer;
    rArea: real;
begin
    //Provided code

    iDiagA := StrToInt(InputBox('Diagonal one of kite(cm)', 'Enter the
value of diagonal one (A): ', '20'));
    iDiagB := StrToInt(InputBox('Diagonal two of kite (cm)',
    'Enter the value of diagonal two (B):', ''));

// Question 1.2
    if iDiagA > iDiagB then
    begin
        rArea := (iDiagA * iDiagB) / 2;
        lblQ1_2.Caption := 'The area of the kite is ' + FloatToStrF
            (rArea, ffFixed, 10, 1) + ' square cm.';
    end
    else
    begin
        ShowMessage('The value of diagonal two (B) must be less than the
            value of diagonal one (A)');
    end;
end;

//=====
//
//                                     Question 1.3 - 11 marks
//=====

procedure TfrmQuestion1.btnQ1_3Click(Sender: TObject);
// Provided code
var
    iNumber, iRemainder: integer;
    sBinary: String;
begin
    // Question 1.3
    iNumber := StrToInt(edtQ1_3.Text);
    sBinary := '';
    while iNumber <> 0 do
    begin
        iRemainder := iNumber MOD 2;
        iNumber := iNumber DIV 2;
        sBinary := IntToStr(iRemainder) + sBinary;
    end;
    lblQ1_3.Caption := 'Binary number: ' + sBinary;
end;
```

```
//=====
//
//                                Question 1.4 – 17 marks
//=====
procedure TfrmQuestion1.btnQ1_4Click(Sender: TObject);
var
    sWord: String;
    iTotal, iValue, iRand, iLen, iCount: integer;
    cChar: char;
begin
    // Provided code
    redQ1_4.Clear;
    // Question 1.4
    iTotal := 0;
    sWord := Uppercase(edtQ1_4.Text);

    if pos(' ', sWord) = 0 then
    begin
        iLen := Length(sWord);

        for iCount := 1 to iLen do
        begin
            cChar := sWord[iCount];

            if cChar in ['A', 'E', 'I', 'O', 'U'] then
                iTotal := iTotal + 3
            else
                if cChar in ['A'..'Z'] then
                    iTotal := iTotal + 2
                else
                    iTotal := iTotal + 1;
            end;

            redQ1_4.lines.Add('Total number of points: ' + IntToStr(iTotal));
        end
    else
    begin
        edtQ1_4.Clear;
        edtQ1_4.SetFocus;
        ShowMessage('Disqualified - more than one word was entered.');
```

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
    TfrmQuestion2 = class(TForm)
        pnlBtns: TPanel;
        btnRestoreDB: TBitBtn;
        grpTblHourLogs: TGroupBox;
        grpTblEmployees: TGroupBox;
        dbgEmployees: TDBGrid;
        dbgHourLogs: TDBGrid;
        tabsQ2_2ADO: TTabSheet;
        tabsQ2_1SQL: TTabSheet;
        btnQ2_2_1: TButton;
        redQ2: TRichEdit;
        grpresults: TGroupBox;
        dbgSQL: TDBGrid;
        grpOutput: TGroupBox;
        pgcTabs: TPageControl;
        pnlTables: TPanel;
        btnQ2_1_1: TButton;
        btnQ2_1_2: TButton;
        btnQ2_1_3: TButton;
        btnQ2_1_4: TButton;
        btnQ2_1_5: TButton;
        btnQ2_2_3: TButton;
        btnQ2_2_2: TButton;
        procedure btnRestoreDBClick(Sender: TObject);
        procedure FormCreate(Sender: TObject);
        procedure FormClose(Sender: TObject; var Action: TCloseAction);
        procedure btnQ2_1_1Click(Sender: TObject);
        procedure btnQ2_1_2Click(Sender: TObject);
        procedure btnQ2_1_3Click(Sender: TObject);
        procedure btnQ2_1_4Click(Sender: TObject);
        procedure btnQ2_1_5Click(Sender: TObject);
        procedure btnQ2_2_1Click(Sender: TObject);
        procedure btnQ2_2_3Click(Sender: TObject);
        procedure btnQ2_2_2Click(Sender: TObject);

    private
    public
    end;

var
    frmQuestion2: TfrmQuestion2;
    dbCONN: TConnection;

```

```
// --- Global variables provided ---
tblEmployees, tblHourLogs: TADOTable;
qryDB: TADOQuery;
implementation
{$R *.dfm}
{$R+}
```

Question 2.1 – SQL section

```
//=====
//
//                                     Question 2.1.1 – 4 marks
//=====
```

```
procedure TfrmQuestion2.btnQ2_1_1Click(Sender: TObject);
var
    sSQL1: String;
begin
    sSQL1 := 'SELECT * FROM tblEmployees ' +
              'ORDER BY JobTitle, HourlyWage DESC';

    // Provided code – do not change
    dbCONN.DBExtract(sSQL1);
end;
```

```
//=====
//
//                                     Question 2.1.2 – 4 marks
//=====
```

```
procedure TfrmQuestion2.btnQ2_1_2Click(Sender: TObject);
var
    sSQL2: String;
begin
    sSQL2 := 'SELECT EmployeeID, LastName, FirstName FROM tblEmployees ' +
              'WHERE JobTitle LIKE "%Engineer%";

    // Provided code – do not change
    dbCONN.DBExtract(sSQL2);
end;
```

```
//=====
//
//                                     Question 2.1.3 – 3 marks
//=====
```

```
procedure TfrmQuestion2.btnQ2_1_3Click(Sender: TObject);
var
    sSQL3: String;
begin
    sSQL3 := 'SELECT DISTINCT JobTitle FROM tblEmployees';

    // Provided code – do not change
    dbCONN.DBExtract(sSQL3);
end;
```

```
//=====
//                                     Question 2.1.4 - 2 marks
//=====
procedure TfrmQuestion2.btnQ2_1_4Click(Sender: TObject);
var
    sSQL4: String;
    bChange: boolean;
begin
    sSQL4 := 'DELETE FROM tblHourLogs ' +
            'WHERE HoursWorked = 99';

    // Provided code - do not change
    dbCONN.ExecuteSQL(sSQL4, dbgHourLogs);
    if bChange then
        begin
            MessageDlg('Database updated', mtInformation, [mbOK], 0);
        end;
end;

//=====
//                                     Question 2.1.5 - 9 marks
//=====
procedure TfrmQuestion2.btnQ2_1_5Click(Sender: TObject);
var
    sSQL5: String;
begin
    sSQL5 := 'SELECT LastName, FORMAT(SUM((HoursWorked - 8) * (HourlyWage
* 2)), "CURRENCY") ' +
            'AS OvertimeAmt ' +
            'FROM tblEmployees E, tblHourLogs H ' +
            'WHERE E.EmployeeID = H.EmployeeID ' +
            'GROUP BY LastName';

    // Provided code - do not change
    dbCONN.DBExtract(sSQL5);
end;
```

Question 2.2 - Delphi section

```
//=====
//
//                               Question 2.2.1 - 7 marks
//=====
procedure TfrmQuestion2.btnQ2_2_1Click(Sender: TObject);
var
    iCount: integer;
begin
    // Provided code
    redQ2.Clear;
    redQ2.Paragraph.TabCount := 3;
    redQ2.Paragraph.Tab[0] := 100;
    redQ2.Paragraph.Tab[1] := 190;
    redQ2.SelAttributes.Style := [fsBold, fsUnderline];
    redQ2.Lines.Add('EmployeeID' + #9 + 'LastName' + #9 + 'JobTitle');

    // Enter your code here for Question 2.2.1
    iCount := 0;
    tblEmployees.First;
    while tblEmployees.Eof = False do
    begin
        if tblEmployees['FirstAidTraining'] = True then
        begin
            redQ2.Lines.Add(tblEmployees['EmployeeID'] + #9 +
                           tblEmployees['LastName'] + #9 +
                           tblEmployees['JobTitle']);

            Inc(iCount);
        end;
        tblEmployees.Next;
    end;
    redQ2.Lines.Add(#10 + 'Total number of employees with first aid
    training: ' + IntToStr(iCount));
end;

//=====
//
//                               Question 2.2.2 - 6 marks
//=====
procedure TfrmQuestion2.btnQ2_2_2Click(Sender: TObject);
begin
    // Question 2.2.2
    tblEmployees.Insert();
    tblEmployees['EmployeeID'] := 'EMP876';
    tblEmployees['FirstName'] := 'Robert';
    tblEmployees['LastName'] := 'Laubscher';
    tblEmployees['HourlyWage'] := 195.00;
    tblEmployees['JobTitle'] := 'Marine Engineer';
    tblEmployees['FirstAidTraining'] := True;
    tblEmployees.Post();
end;
```

```
//=====
//
//                               Question 2.2.3 - 5 marks
//=====
procedure TfrmQuestion2.btnQ2_2_3Click(Sender: TObject);
var
    iHours: Integer;
begin
    // Question 2.2.3
    iHours := StrToInt(edtQ2_2_3.text);
    tblHourLogs.Edit();
    tblHourLogs['HoursWorked'] := iHours;
    tblHourLogs.Post();
end;

//=====
//                               Setup DB connections - DO NOT CHANGE!
//=====
{$REGION DB CONNECTION}
// =====
procedure TfrmQuestion2.btnRestoreDBCClick(Sender: TObject);
begin
    // Restores the Database
    dbCONN.RestoreDatabase(dbgEmployees, dbgHourLogs, dbgSQL);
    redQ2.Clear;
    // Formatting field datatypes
    tblEmployees := dbCONN.tblOne;
    tblHourLogs := dbCONN.tblMany;
    qryDB := dbCONN.qry;
end;

// =====
procedure TfrmQuestion2.FormClose(Sender: TObject; var Action:
TCloseAction);
begin // Disconnects from database and closes all open connections
    dbCONN.dbDisconnect;
end;

// =====
procedure TfrmQuestion2.FormCreate(Sender: TObject);
begin
    CurrencyString := 'R';
    ShortDateFormat := 'YYYY/MM/DD';
    // Sets up the connection to database and opens the tables.
    dbCONN := TConnection.Create;
    dbCONN.dbConnect;
    tblEmployees := dbCONN.tblOne;
    tblHourLogs := dbCONN.tblMany;
    qryDB := dbCONN.qry;
    dbCONN.SetupGrids(dbgEmployees, dbgHourLogs, dbgSQL);
    pgcTabs.ActivePageIndex := 0;
end;
// =====
{$ENDREGION}

end.
```

ANNEXURE G: SOLUTION FOR QUESTION 3**OBJECT CLASS UNIT:**

```

unit Transaction_U;

interface

type
  TTransaction = class(TObject)
  private
    var
      fCustomerID: String;
      fContainerSize: char;
      fStoragePeriod: integer;
      fAmountPaid: real;
  public
    constructor create(sCustomerID: String; cContainerSize: char;
      iStoragePeriod: integer);
    function getAmountPaid: real;
    procedure updateAmountPaid(pAmountPaid: real);
    function calculateCost: real;
    function toString: String;
  end;

implementation

{ TTransaction }

uses
  SysUtils, Math;

//=====
//                               Question 3.1.1 - 4 marks
//=====
constructor TTransaction.create(sCustomerID: String; cContainerSize:
      char; iStoragePeriod: integer);
begin
  fCustomerID := sCustomerID;
  fContainerSize := cContainerSize;
  fStoragePeriod := iStoragePeriod;
  fAmountPaid := 0.00;
end;

//=====
//                               Question 3.1.2 - 2 marks
//=====

function TTransaction.getAmountPaid: real;
begin
  result := fAmountPaid;
end;

```

```
//=====
//                                     Question 3.1.3 – 3 marks
//=====
```

```
procedure TTransaction.updateAmountPaid(pAmountPaid: real);
begin
    fAmountPaid := fAmountPaid + pAmountPaid;
end;
```

```
//=====
//                                     Question 3.1.4 – 11 marks
//=====
```

```
function TTransaction.calculateCost: real;
var
    rCost, rPercentageDiscount: real;
begin
    case fContainerSize of
        'S': rCost := 1000.00;
        'M': rCost := 1750.00;
        'L': rCost := 2500.00;
    end;

    rCost := fStoragePeriod * rCost;
    rPercentageDiscount := Floor(fStoragePeriod / 6) * 0.1;
    if rPercentageDiscount > 0.5 then
    begin
        rPercentageDiscount := 0.5;
    end;
    result := rCost - (rCost * rPercentageDiscount);
end;
```

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

```
function TTransaction.toString: String;
begin
    result := 'Customer ID: ' + fCustomerID + #10 +
        'Container size: ' + fContainerSize + #10 +
        'Storage period (months): ' + IntToStr(fStoragePeriod) + #10
        + 'Amount paid: ' + Format('%.2m', [fAmountPaid]);
end;

end.
```

MAIN CLASS UNIT:

```

unit Question3_U;

interface

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

type
  TQuestion3 = class(TForm)
    gbxQ3_2_1: TGroupBox;
    gbxContainerSize: TGroupBox;
    lstQ3_2_1: TListBox;
    gbxCustomerID: TGroupBox;
    edtQ3_2_1: TEdit;
    gbxNumberOfMonths: TGroupBox;
    spnQ3_2_1: TSpinEdit;
    gbxQ3_2_3: TGroupBox;
    redQ3_2: TRichEdit;
    btnQ3_2_1: TButton;
    btnQ3_2_3: TButton;
    btnReset: TButton;
    gbxQ3_2_2: TGroupBox;
    btnQ3_2_2_b: TButton;
    gbxAmount: TGroupBox;
    edtQ3_2_2: TEdit;
    pnlQ3_2_2: TPanel;
    btnQ3_2_2_a: TButton;
    procedure btnQ3_2_1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure btnQ3_2_3Click(Sender: TObject);
    procedure btnResetClick(Sender: TObject);
    procedure btnQ3_2_2_bClick(Sender: TObject);
    procedure btnQ3_2_2_aClick(Sender: TObject);
  private
  public

  end;

var
  Question3: TQuestion3;

implementation

{$R *.dfm}

uses
  Transaction_U;

var
  objTransaction: TTransaction;

```

```
//=====
//
//                                     Question 3.2.1 – 7 marks
//=====
```

```
procedure TQuestion3.btnQ3_2_1Click(Sender: TObject);
var
    sCustomerID: String;
    cContainerSize: char;
    iStoragePeriod: integer;
begin
    // Question 3.2.1
    sCustomerID := edtQ3_2_1.Text;
    cContainerSize := lstQ3_2_1.Items[lstQ3_2_1.ItemIndex][1];
    iStoragePeriod := spnQ3_2_1.Value;

    objTransaction := TTransaction.create(sCustomerID,cContainerSize,
                                          iStoragePeriod);

    btnQ3_2_1.Enabled := False;
end;
```

```
//=====
//
//                                     Question 3.2.2(a) – 3 marks
//=====
```

```
procedure TQuestion3.btnQ3_2_2_aClick(Sender: TObject);
begin
    // Question 3.2.2 (a)
    pnlQ3_2_2.Caption := 'Amount due: ' + Format('%.2m',
[objTransaction.calculateCost]);
end;
```

```
//=====
//
//                                     Question 3.2.2(b) – 5 marks
//=====
```

```
procedure TQuestion3.btnQ3_2_2_bClick(Sender: TObject);
var
    rAmountPaid: real;
begin
    // Question 3.2.2 (b)
    rAmountPaid := StrToFloat(edtQ3_2_2.Text);

    objTransaction.updateAmountPaid(rAmountPaid);

    pnlQ3_2_2.Caption := 'Amount due: ' + Format('%.2m',
[objTransaction.calculateCost - objTransaction.getAmountPaid]);
end;
```

```
//=====
//                                     Question 3.2.3 – 2 marks
//=====
```

```
procedure TQuestion3.btnQ3_2_3Click(Sender: TObject);
begin
    // Provided code
    redQ3_2.Clear;
    // Question 3.2.3
    redQ3_2.Lines.Add(objTransaction.toString);
end;
```

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

```
procedure TQuestion3.FormCreate(Sender: TObject);
begin
    lstQ3_2_1.ItemIndex := 0;
end;
```

```
procedure TQuestion3.btnResetClick(Sender: TObject);
begin
    btnQ3_2_1.Enabled := True;
end;
```

```
end.
```

ANNEXURE H: SOLUTION FOR QUESTION 4

```

unit Question4_U;

interface

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

type
  TfrmQuestion4 = class(TForm)
    GroupBox1: TGroupBox;
    redQ4_1: TRichEdit;
    btnQ4_1_2: TButton;
    GroupBox2: TGroupBox;
    GroupBox3: TGroupBox;
    Image1: TImage;
    redQ4_2: TRichEdit;
    pnlQ4: TPanel;
    btnQ4_2: TButton;
    btnQ4_1_1: TButton;
    procedure btnQ4_2Click(Sender: TObject);
    procedure btnQ4_1_2Click(Sender: TObject);
    procedure btnQ4_1_1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);

  private
    { Private declarations }
    tFile : TextFile;
    arrContainers : array [1..50] of real;
    // arrShip    : array [1..5, 1..10] of real; optional
  public
    { Public declarations }
  end;
//=====
// Provided code - DO NOT CHANGE
//=====

const
  arrTempContainers : array[1..50] of real =
    (31.2, 43.4, 5.1, 41.2, 52.6, 41.9, 97, 49.3, 15.8, 39.5,
     78.8, 96.3, 67.8, 47.2, 31.4, 18.4, 27, 1.4, 33.5, 16.5,
     58.3, 9.9, 62.9, 11.8, 62.5, 59, 13.4, 49.8, 60.4, 74.5,
     13.5, 67.7, 94, 39.4, 47.4, 13.1, 26.2, 63, 89, 22.3,
     54, 16.9, 38.6, 46.2, 22.5, 11.4, 65.1, 48.6, 41.4, 47.9);

var
  frmQ4: TfrmQ4;

implementation

{$R *.dfm}

```

```
//=====
//
//                                Question 4.1.1 - 4 marks
//=====

procedure TfrmQuestion4.btnQ4_1_1Click(Sender: TObject);
var
    iRow : integer;
begin
    // Provided code
    btnQ4_2.Enabled := false;
    redQ4_2.Clear;
    redQ4_1.Clear;
    pnlQ4.Caption := ' ' ;

    // Question 4.1.1

    for iRow := 1 to 50 do
        arrContainers[iRow] := RoundTo(Random() + Random(99), -1);

    btnQ4_1_2.Enabled := true;
    //for iRow := 1 to 50 do
    //    arrContainers[iRow] := arrTempContainers[iRow];

end;
//=====
//
//                                Question 4.1.2 - 8 marks
//=====

procedure TfrmQuestion4.btnQ4_1_2Click(Sender: TObject);
var
    iRow, iCol, iIndex : integer;
    sLine : String;

begin
    // Provided code
    redQ4_1.Clear;
    btnQ4_2.Enabled := true;

    // Question 4.1.2
    iIndex := 0;
    for iRow := 1 to 5 do
        begin
            sLine := '';
            for iCol := 1 to 10 do
                begin
                    Inc(iIndex);
                    sLine := sLine + FloatToStr(arrContainers[iIndex]) + #9;
                end;
            redQ4_1.Lines.Add(sLine);
        end;
    end;
end;
```

```
//=====
//                                     Question 4.2 – 18 marks
//=====

procedure TfrmQuestion4.btnQ4_2Click(Sender: TObject);
var
  rTotalTons : real;
  iRow,iCol,iIndex : integer;
  sLine : String;
begin
  // Provided code
  redQ4_2.Clear;

  //Question 4.2

  rTotalTons := 0;
  iIndex := 1;
  sLine := '';
  while (iIndex <= 50) and (rTotalTons <= 200) do
    begin
      rTotalTons := rTotalTons + arrContainers[iIndex];
      if (rTotalTons > 200) then
        rTotalTons := rTotalTons - arrContainers[iIndex]
      else
        sline := sLine + FloatToStr(arrContainers[iIndex])+#9 ;

        Inc(iIndex);
      end;
    end;

  // Alternative
  iRow := 1;
  iIndex := 1;    //one dim array
  sLine := '';

  while (rTotalTons < 200) and (iRow < 5) do
    begin
      iCol := 1;
      while (iCol < 11) and (iIndex <= 50) do
        begin
          arrShip[iRow,iCol] := arrContainers[iIndex];
          if rTotalTons + arrShip[iRow,iCol] <= 200 then //rTotalTons +
            arrContainers[iIndex] <= 200
          begin
            sline := sLine + FloatToStr(arrShip[iRow,iCol])+#9 ;
            rTotalTons := rTotalTons + arrShip[iRow,iCol];
            Inc(iCol);
          end;
          Inc(iIndex);
        end;
      Inc(iRow);
    end;
  AssignFile(tFile, 'Tons.txt');
  Rewrite(tFile);
  writeln(tfile, sline);
  CloseFile(tFile);
  redQ4_2.Lines.Add(sLine);
```

```
pnlQ4.Caption := 'Total weight of load: ' + FloatToStr (rTotalTons );  
  
end;  
  
// Provided code  
procedure TfrmQuestion4.FormCreate(Sender: TObject);  
begin  
    btnQ4_1_2.Enabled := false;  
    btnQ4_2.Enabled := false;  
end;  
  
end.
```