



basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

NOVEMBER 2025

MARKING GUIDELINES

MARKS: 150

These marking guidelines consist of 27 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 12) include the marking grid for each question.
- **Annexures E, F, G and H** (pages 13 to 27) contain examples of solutions for Questions 1 to 4 in programming code.
- Copies of **Annexures A, B, C, D** and **the summary for the marks of the learner** (pages 3 to 12) 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 - Increase value] Extract the number from pnlQ1_1 ✓ Convert to integer ✓ Increase the number by 1 ✓ Display the number in pnlQ1_1 converted to string ✓	4	
1.2	Button [1.2 - Volume] Extract the length of the one side from spnQ1_2 ✓ Calculate volume: rVolume := 5 * ✓ (3 + Sqrt(5)) ✓ / 12 ✓ * Power(iSide,3) ✓ Display volume in edtQ1_2 converted to a string and formatted to 2 decimal places ✓ NOTE: Any TWO mathematical functions can be used.	6	
1.3	Button [1.3 - Shopping aisle] Extract the code from edtQ1_3 ✓ Find the position of hash (#) (or other mechanism to separate the aisle number from the category) ✓ Extract aisle number ✓ Extract/use the category as character ✓ Use a Case correctly ✓ For all possible characters ✓ ('F', 'D', 'B', 'S', 'P') ✓ to determine category description ✓ Compile a string with "Aisle ", aisle number ✓, ": " and category description ✓ Display the string in lblQ1_3 ✓	11	

1.4	Button [1.4 - Populate and display] Loop from 1 to length of array ✓ Generate random numbers ✓ in correct range (1 – 15) ✓ Assign the random number to each element in the array ✓ Concatenate the array values to build the string ✓ separating the values with a dash (-) ✓ Mechanism to deal with extra dash ✓ Display the concatenated string in memQ1_4 ✓	8	
1.5	Button [1.5 - Determine HCF] Test IF Num1 < Num2 ✓ Assign Num1 to LowerNum ✓ Else Assign Num2 to LowerNum ✓ <i>//Use of the While loop</i> Cnt = 1; ✓ While Cnt <= LowerNum ✓ Test IF ✓ Num1 MOD Cnt = 0 ✓ Test IF Num2 MOD Cnt = 0 ✓ HCF = Cnt ✓ Inc(Cnt) ✓ Display HCF in edtQ1_5 ✓ <i>//Alternative using the FOR... loop</i> Loop Cnt (1) from 1 to (1) LowerNum (1) Test IF (1) Num1 MOD Cnt = 0 (1) Test IF Num2 MOD Cnt = 0 (1) HCF = Cnt (1) Display HCF in edtQ1_5 (1)	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 - Farm details] SELECT FarmName, NearestTown, SizeInHectares ✓ FROM tblFarms ✓ ORDER BY SizeInHectares DESC ✓	3	
2.1.2	Button [2.1.2 - Contact details] SELECT FullName, ContactNumber FROM tblFarmOwners ✓ WHERE Email IS NULL ✓ Alternative: ISNULL(Email)		
2.1.3	Button [2.1.3 - Average farm size] SELECT FarmType, FORMAT(AVG(SizeInHectares) ✓, "0.00" ✓) AS AverageSize ✓ FROM tblFarms WHERE FarmType = ✓ ' ' + sFarmType + ' ' ✓ GROUP BY FarmType ✓ Alternative: FarmType = ' + QuotedStr(sFarmType) + ' FarmType LIKE ' ' + sFarmType + ' '	6	
2.1.4	Button [2.1.4 - Young farm owners] SELECT Fullname, DateOfBirth, ✓ INT ✓ ((NOW ✓ - DateOfBirth) /365 ✓) AS Age FROM tblFarmOwners ✓ WHERE INT((NOW - DateOfBirth)/365) ✓ <= 25 ✓ Alternative: (Date() - DateOfBirth)		

2.1.5	Button [2.1.5 - Multiple farms in KZN]	7	
	<pre> SELECT FullName, COUNT(*) ✓ AS TotalFarms FROM tblFarmOwners, tblFarms ✓ WHERE tblFarmOwners.OwnerID = tblFarms.OwnerID ✓ AND ✓ (NearestTown = "Durban" OR NearestTown = "Pietermaritzburg") ✓ GROUP BY FullName ✓ HAVING COUNT(*) > 1 ✓ Alternative: NearestTown IN ("Durban", "Pietermaritzburg") </pre>		
	Subtotal:	25	

QUESTION 2: MARKING GRID (CONT.)

2.2	Database Manipulation		
	Button [2.2 - Mixed-farm type] Test IF file exists = False/try..except ✓ Display a suitable message ✓ Exit or else statement ✓ AssignFile(tFile, 'MixedFarms.txt') ✓ Reset(tFile) ✓ Loop through Text file ✓ Readln(tFile, FarmID variable) ✓ tblFarms.First ✓ Loop through tblFarms ✓ Test IF FarmID variable = tblFarms['FarmID'] ✓ Test IF tblFarms['SizeInHectares'] > 100 ✓ tblFarms.Edit ✓ tblFarms['FarmType'] = 'Mixed' ✓ tblFarms.Post Else Display tblFarms['FarmID'] as string, tblFarms['FarmName'] and tblFarms['SizeInHectares'] as string in redQ2_2 ✓ tblFarms.Next ✓ End loop (tblFarms) End loop (Text file)	15	
	Subtotal:	15	
	TOTAL SECTION B:	40	

ANNEXURE C**QUESTION 3: MARKING GRID – OBJECT-ORIENTATED PROGRAMMING**

CENTRE NUMBER:		EXAMINATION NUMBER:	
QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
3.1.1	Constructor method Constructor heading with String, Integer and Boolean parameters ✓ Assign the three parameters to correct attributes ✓ Assign the following default values to the other attributes: fNoOfHarvests = 2 fTotalHoneyHarvested = 80 fHarvestDates = '2025/01/05' + #13 + '2025/06/23' ✓	4	
3.1.2	Function getNoOfHarvests Function heading getNoOfHarvests with integer return type ✓ Return fNoOfHarvests ✓	2	
3.1.3	Function calcAverage Function heading calcAverage ✓ with real return type ✓ Return ✓ fTotalHoneyHarvested/ fNoOfHarvests ✓ ALSO ACCEPT: fTotalHoneyHarvested/ getNoOfHarvests	4	
3.1.4	Procedure updateBeehiveDetails Procedure heading with a real parameter ✓ fTotalHoneyHarvested = fTotalHoneyHarvested ✓ + parameter ✓ fNoOfHarvests = fNoOfHarvests + 1 ✓ fHarvestDates = fHarvestDates ✓ + #13 + system date ✓	6	

3.1.5	<p>Function checkHealthStatus</p> <p>Function heading with Boolean return type ✓ Test IF ((fBeeCount > 7000) ✓ OR (fNoOfHarvests <=3)) ✓ AND (fPests = False) ✓</p> <p> Result = True ✓ Else Result = False ✓</p> <p><i>Alternative 1:</i> Function heading with Boolean return type (1) Result = False (1) Test IF (fBeeCount > 7000) (1) OR (fNoOfHarvests <=3) (1) Test IF (fPests = False) (1) Result = True (1)</p> <p><i>Alternative 2:</i> Function heading with Boolean return type (1) Result = (2) ((fBeeCount > 7000) (1) OR (fNoOfHarvests <=3)) (1) AND (fPests = False) (1)</p>	6	
	Subtotal: Object class	22	

QUESTION 3: MARKING GRID (CONT.)

QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
3.2.1	Button [3.2.1 - Instantiate beehive object] objBeehive := ✓ TBeehive.Create() ✓ Extracted parameters in sequence ✓ Display object in redQ3_2 ✓ using toString method ✓	5	
3.2.2	Button [3.2.2 - Ready to harvest?] Test IF objBeehive.checkHealthStatus = TRUE ✓ Display 'Ready to harvest' on lblQ3_2_2 btnQ3_2_3.Enabled = True Else ✓ Display 'Not ready to harvest' on lblQ3_2_2 ✓ //both btnQ3_2_3.Enabled = False ✓ //both Concepts: Test IF condition (1) with Else statement (1) Display message 'Ready to harvest'/ 'Not ready to harvest' in lblQ3_2_2 (1) btnQ3_2_3.Enabled = True/False(1)	4	
3.2.3	Button [3.2.3 - Honey harvested] Extract Value from edtQ3_2_3 as real data type ✓ Call objBeehive.updateBeehiveDetails (Value) ✓ Display in redQ3_2: Label + objBeehive.getNoOfHarvests as string ✓ objBeehive.toString ✓	4	
3.2.4	Button [3.2.4 - Average honey harvested] Call objBeehive.calcAverage ✓ to display average using ShowMessage ✓ formatted to 2 decimal places ✓	3	
3.2.5	Button [3.2.5 - Change status of pests] Call objBeehive.setPests ✓ Disable btnQ3_2_3 ✓	2	
	Subtotal Form class:	18	
	TOTAL SECTION C:	40	

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

CENTRE NUMBER:		EXAMINATION NUMBER:	
QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
4.1.1	Button [4.1.1 - Extract] Loop Row from 1 to 10 ✓ Loop Col from 1 to 2 ✓ Find position of placeholder ✓ in arrSoil[Row] ✓ Extract value as number ✓ and store in arr2DSoil[Row,Col] ✓ Delete from index 1 up to placeholder ✓ Mechanism to deal with last value ✓	8	
4.1.2	Button [4.1.2 - Validate] Loop Row from 1 to 10 ✓ Initialise Total to 0 ✓ Loop Col from 1 to 3 ✓ Increment Total with value of arr2DSoil[Row,Col] ✓ Test IF Total < 100 ✓ Loop A from 1 to 3 ✓ Calculate ratio arr2DSoil[Row, A] ✓ / Total *100 ✓ Display hectare with Row number as string in redQ4_1 ✓	9	

4.2	Button [4.2 - Nutrient markers] Loop Loop1 from 1 to 50000 ✓ Initialise Sum to 0 ✓ Loop ✓ Loop2 from 1 to Length(Loop1) ✓ //number of digits Initialise Factorial to 1 ✓ Loop ✓ Loop3 from 1 to typecast integer (typecast string (Loop1)✓ [Loop2]) ✓ //value of digit Factorial = Factorial * Loop3✓ Sum = Sum + Factorial ✓ Test ✓ IF Sum = Loop1 ✓ Display Loop1/ Sum as string in memQ4_2 ✓ Concepts: Outer Loop1 (1) Initialise Sum variable (1) Inner Loop2 - Nested loop2 (1) using length of the string of outer loop variable (1) Initialise Factorial variable (1) Digit extraction - Nested Loop3 (1), Digit extraction (1), Typecast to integer (1) Factorial calculation (1) Sum logic (1) Comparison logic - IF statement (1) Sum = outer loop1 (1) Output (1)	13	
	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

```
unit Question1_U;

interface

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

type
  gpbQ1_1: TGroupBox;
  pnlQ1_1: TPanel;
  btnQ1_1: TButton;
  gpbQ1_2: TGroupBox;
  lblLength: TLabel;
  spnQ1_2: TSpinEdit;
  imgIcosahedron: TImage;
  edtQ1_2: TEdit;
  btnQ1_2: TButton;
  gpbQ1_3: TGroupBox;
  lblDate: TLabel;
  edtQ1_3: TEdit;
  lblQ1_3: TLabel;
  gpbQ1_4: TGroupBox;
  memQ1_4: TMemo;
  btnQ1_4: TButton;
  lblVolume: TLabel;
  btnQ1_3: TButton;
  GroupBox1: TGroupBox;
  edtQ1_5_Num1: TEdit;
  Label1: TLabel;
  Label2: TLabel;
  edtQ1_5_Num2: TEdit;
  btnQ1_5: TButton;
  edtQ1_5: TEdit;
  procedure btnQ1_1Click(Sender: TObject);
  procedure btnQ1_2Click(Sender: TObject);
  procedure btnQ1_3Click(Sender: TObject);
  procedure btnQ1_4Click(Sender: TObject);
  procedure btnQ1_5Click(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);  
var  
    iNum: Integer;  
begin  
  
    // Question 1.1  
    iNum := StrToInt(pnlQ1_1.Caption);  
    Inc(iNum);  
    pnlQ1_1.Caption := IntToStr(iNum);  
end;  
  
// =====  
// Question 1.2           6 marks  
// =====  
procedure TfrmQuestion1.btnQ1_2Click(Sender: TObject);  
var  
    iSide: Integer;  
    rVolume: Real;  
begin  
    // Question 1.2  
    iSide := spnQ1_2.Value;  
    rVolume := 5 * (3 + Sqrt(5)) / 12 * Power(iSide,3);  
    edtQ1_2.Text := FloatToStrF(rVolume, ffFixed, 8, 2);  
end;  
  
// =====  
// Question 1.3           11 marks  
// =====  
procedure TfrmQuestion1.btnQ1_3Click(Sender: TObject);  
var  
    sCode, sOutput: string;  
    sAisle: string;  
    cCategory : Char;  
    iPos : Integer;  
begin  
    // Question 1.3  
    sCode := edtQ1_3.Text;  
    iPos := Pos('#', sCode);  
    cCategory := sCode[iPos+1];  
    sAisle := Copy(sCode, 1, iPos-1);  
    case cCategory of  
        'F': sOutput := 'Fruit and vegetables';  
        'D': sOutput := 'Dairy';  
        'B': sOutput := 'Butchery';  
        'S': sOutput := 'Sauces';  
        'P': sOutput := 'Pasta and Rice';  
    end;  
    sOutput := 'Aisle ' + sAisle + ': ' + sOutput;  
    lblQ1_3.Caption := sOutput;  
end;
```

```
// =====
// Question 1.4    8 marks
// =====
procedure TfrmQuestion1.btnQ1_4Click(Sender: TObject);
//Provided code
var
    arrNumbers: array [1 .. 9] of Integer;
//End of provided code

    iCnt: Integer;
    sOut: String;
begin
    // Question 1.4

    sOut := '';
    for iCnt := 1 to length(arrNumbers) do
    begin
        arrNumbers[iCnt] := RandomRange(1, 16);
        sOut := sOut + '-' + IntToStr(arrNumbers[iCnt]);
    end;
    Delete(sOut, length(sOut), 1);
    memQ1_4.Text := sOut;
end;

// =====
// Question 1.5    11 marks
// =====
procedure TfrmQuestion1.btnQ1_5Click(Sender: TObject);
var
    iCnt: Integer;
    iHCF, iLowerNum, iNum1, iNum2: Integer;
begin
    //Provided code
    iNum1 := StrToInt(edtQ1_5_Num1.Text);
    iNum2 := StrToInt(edtQ1_5_Num2.Text);
    iHCF := 0;

    // Question 1.5
    // Code using a while loop
    if iNum1 < iNum2 then
        iLowerNum := iNum1
    else
        iLowerNum := iNum2;

    // Code using a while loop
    iCnt := 1;
    While iCnt <= iLowerNum do
    begin
        if (iNum1 MOD iCnt = 0) then
            if (iNum2 MOD iCnt = 0) then
                iHCF := iCnt;
            Inc(iCnt);
        end;
    end;
```

Code using a Repeat Until loop

```
{ iCnt := 1;
  Repeat
    if (iNum1 MOD iCnt = 0) then
      if (iNum2 MOD iCnt = 0) then
        iHCF := iCnt;
      Inc(iCnt);
    Until iCnt > iLowerNum;
  end;}
```

//Code using a For loop

```
{for iCnt := 1 to iLowerNum do
begin
  if (iNum1 MOD iCnt = 0) then
    if (iNum2 MOD iCnt = 0) then
      begin
        iHCF := iCnt;
      end;
end;}
```

```
edtQ1_5.Text := 'HCF: ' + IntToStr(iHCF);
```

```
end;
```

```
end.
```


ANNEXURE F: SOLUTION FOR QUESTION 2

```
//=====
// Question 2.1 – Section: SQL statements
//=====

//=====
// Question 2.1.1          3 marks
//=====
procedure TfrmQuestion2.btnQ2_1_1Click(Sender: TObject);
var
    sSQL1: string;
begin
    // Question 2.1.1

    sSQL1 := 'SELECT FarmName, NearestTown, SizeInHectares FROM tblFarms ' +
        'ORDER BY SizeInHectares DESC';

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

//=====
// Question 2.1.2          2 marks
//=====
procedure TfrmQuestion2.btnQ2_1_2Click(Sender: TObject);
var
    sSQL2: string;
begin
    // Question 2.1.2

    sSQL2 := 'SELECT FullName, ContactNumber ' +
        'FROM tblFarmOwners WHERE Email IS NULL';

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

//=====
// Question 2.1.3          6 marks
//=====
procedure TfrmQuestion2.btnQ2_1_3Click(Sender: TObject);
var
    sSQL3, sFarmType: string;
begin
    // Provided code
    sFarmType := cmbQ2_1_3.Text;

    // Question 2.1.3
    sSQL3 :=
        'SELECT FarmType, FORMAT(AVG(SizeInHectares),"0.00") as AverageSize ' +
        'FROM tblFarms ' + 'WHERE FarmType = "' + sFarmType + '" GROUP BY
        FarmType';

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

```
//=====
// Question 2.1.4           7 marks
//=====
procedure TfrmQuestion2.btnQ2_1_4Click(Sender: TObject);
var
    sSQL4: string;
begin
    //Question 2.1.4

    sSQL4 := 'SELECT FullName, DateOfBirth, INT((NOW - DateOfBirth)/365)
              AS Age ' +
              'FROM tblFarmOwners ' +
              'WHERE INT((NOW - DateOfBirth)/365) <= 25 ';

    // Alternative
    sSQL4:= 'SELECT FullName, INT((NOW - DateOfBirth)/365) AS Age
            ' +
            ' FROM tblFarmOwners' +
            ' WHERE INT((NOW - DateOfBirth)/365) <= 25 ';

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

end;

//=====
// Question 2.1.5           7 marks
//=====
procedure TfrmQuestion2.btnQ2_1_5Click(Sender: TObject);
var
    sSQL5: string;
begin
    // Question 2.1.5
    sSQL5 := 'SELECT  FullName, COUNT(*) AS TotalFarms ' +
              'FROM tblFarmOwners, tblFarms ' +
              'WHERE tblFarmOwners.OwnerID = tblFarms.OwnerID AND
              (NearestTown = "Durban" OR NearestTown = "Pietermaritzburg")' +
              'GROUP BY FullName ' +
              'HAVING COUNT(*) > 1 ';

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

```
//=====
// Question 2.2 – Section Delphi code
//=====

//=====
// Question 2.2          15 marks
// =====

procedure TfrmQuestion2.btnQ2_2Click(Sender: TObject);
var
    tFile: TextFile;
    iFarmID: Integer;
    sFarmType: string;
begin
    // Provided code - do not change
    redQ2_2.Clear;
    redQ2_2.Lines.Add('Farms on list that do not qualify: ');

    // Question 2.2
    Assignfile(tFile, 'MixedFarms.txt');
    try
        Reset(tFile);
    except
        ShowMessage('File does not exist');
        Exit;
    end;

    while not Eof(tFile) do
    begin
        Readln(tFile, iFarmID);
        tblFarms.First;
        while not tblFarms.Eof do
        begin
            if (tblFarms['FarmID'] = iFarmID) then
            begin
                if tblFarms['SizeInHectares'] > 100 then
                begin
                    tblFarms.Edit;
                    tblFarms['FarmType'] := 'Mixed';
                    tblFarms.Post;
                end
            else
                redQ2_2.Lines.Add(IntToStr(tblFarms['FarmID']) + ' - ' + tblFarms
                    ['FarmName'] + ' - ' + IntToStr(tblFarms['SizeInHectares']) +
                    ' hectares');
            end;
            tblFarms.Next;
        end;
    end;

    // Provided code
    tblFarms.Sort := 'FarmID ASC';
end;
```

ANNEXURE G: SOLUTION FOR QUESTION 3**Object class**

```

unit Beehive_U;

interface

Uses SysUtils, DateUtils, Math;

Type
  TBeehive = class(TObject)
  Private
    fBeehiveID: String;
    fBeeCount: Integer;
    fPests: Boolean;
    fNoOfHarvests: Integer;
    fTotalHoneyHarvested: Real;
    fHarvestDates: String;

  Public
    Constructor Create(sBeehiveID: String; iBeeCount: Integer;
                      bPests : boolean);
    Function getNoOfHarvests: Integer;
    Function calcAverage: Real;
    Procedure updateBeehiveDetails(rHoneyKG: Real);
    Function checkHealthStatus: Boolean;

    // Provided code
    Procedure setPests;
    Function toString: String;
  End;
implementation

{ TBeehive }
// =====
// Question 3.1.1           4 marks
// =====
Constructor TBeehive.Create(sBeehiveID: String; iBeeCount: Integer; bPests
: boolean);
begin
  fBeehiveID := sBeehiveID;
  fBeeCount := iBeeCount;
  fPests := bPests;
  fNoOfHarvests := 2;
  fTotalHoneyHarvested := 80;
  fHarvestDates := '2025/01/05' + #13 + '2025/06/23';
end;
// =====
// Question 3.1.2           2 marks
// =====
function TBeehive.getNoOfHarvests: Integer;
begin
  Result := fNoOfHarvests;
end;

```

```
// =====
// Question 3.1.3           4 marks
// =====
function TBeehive.calcAverage: Real;
begin
    Result := fTotalHoneyHarvested / fNoOfHarvests;
end;
// =====
// Question 3.1.4           6 marks
// =====
procedure TBeehive.updateBeehiveDetails(rHoneyKG: Real);
begin
    fTotalHoneyHarvested := fTotalHoneyHarvested + rHoneyKG;
    Inc(fNoOfHarvests);
    fHarvestDates := fHarvestDates + #13 + DateToStr(Date);
end;
// =====
// Question 3.1.5           6 marks
// =====
function TBeehive.checkHealthStatus: Boolean;
var
    bHealthy : Boolean;
begin
    bHealthy := False;
    if (fPests = False) AND ((fBeeCount > 7000) OR
                             (fNoOfHarvests <= 3)) then
        bHealthy := True;
    Result := bHealthy;
//Alternative
    Result:= (fPests = False) AND ((fBeeCount > 7000) OR
                                   (fNoOfHarvests <= 3))
end;
// =====
// Provided methods
// =====
procedure TBeehive.setPest;
begin
    if fPests = True then
        fPests := False
    else
        fPests := True;
end;
function TBeehive.toString: String;
var
    sPestStatus: String;
begin
    if fPestStatus then
        sPestStatus := 'Pests in beehive.'
    else
        sPestStatus := 'No pests in beehive.';
    Result := 'Beehive ID: ' + #9 + fBeehiveID + #13 +
        'Number of bees in beehive: ' + #9 + IntToStr(fBeeCount) + #13 +
        'Honey harvested in kg: ' + #9 +
        FloatToStrF(fTotalHoneyHarvested, ffFixed, 8, 1) + #13 +
        'Number of harvests: ' + #9 + IntToStr(fNoOfHarvests) + #13 + #13 +
        'Harvest dates: ' + #13 + fHarvestDates + #13 + sPestStatus;
end;
end.
```

Main Form Unit

```
unit Question3_U;
interface
uses
    Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
    Forms, Dialogs, StdCtrls, pngimage, ExtCtrls, Spin, ComCtrls, Math,
    beehive_u;

type
    TfrmQuestion3_2 = class(TForm)
        pnlQ3_2: TPanel;
        imgBeeLeft: TImage;
        gpbQ3_2_1: TGroupBox;
        cmbQ3: TComboBox;
        lblCode: TLabel;
        btnQ3_2_1: TButton;
        lblNumBees: TLabel;
        gpbQ3_2_3: TGroupBox;
        lblHavest: TLabel;
        imgBeeRight: TImage;
        gpbQ3_2_4: TGroupBox;
        btnQ3_2_4: TButton;
        redQ3_2: TRichEdit;
        btnQ3_2_3: TButton;
        spnQ3: TSpinEdit;
        edtQ3_2_3: TEdit;
        btnQ3_2_2: TButton;
        gpbQ3_2_2: TGroupBox;
        GroupBox1: TGroupBox;
        lblQ3_2_2: TLabel;
        GroupBox2: TGroupBox;
        GroupBox3: TGroupBox;
        btnQ3_2_5: TButton;
        Edit1: TEdit;
        chbQ3: TCheckBox;
        lblPests: TLabel;
        procedure btnQ3_2_1Click(Sender: TObject);
        procedure FormCreate(Sender: TObject);
        procedure btnQ3_2_3Click(Sender: TObject);
        procedure btnQ3_2_4Click(Sender: TObject);
        procedure btnQ3_2_2Click(Sender: TObject);
        procedure btnQ3_2_5Click(Sender: TObject);
        procedure chbQ3Click(Sender: TObject);
        procedure GroupBox2Click(Sender: TObject);
    private
        { Private declarations }
    public
        { Public declarations }
    end;
var
    frmQuestion3_2: TfrmQuestion3_2;
    // Provided code
    objBeehive: TBeehive;
```

implementation

```
// =====  
// Question 3.2.1           5 marks  
// =====  
procedure TfrmQuestion3_2.btnQ3_2_1Click(Sender: TObject);  
var  
    // Provided code  
    sBeehiveID: string;  
    iNumBees: Integer;  
    bPests: Boolean;  
begin  
    // Provided code  
    redQ3_2.Clear;  
    sBeehiveID := cmbQ3.Text;  
    iNumBees := spnQ3.Value;  
    bPests := chbQ3.checked;  
  
    // Question 3.2.1  
    objBeehive := TBeehive.Create(sBeehiveID, iNumBees, bPests);  
    redQ3_2.Lines.Add(objBeehive.toString);  
end;;  
  
// =====  
// Question 3.2.2           4 marks  
// =====  
procedure TfrmQuestion3_2.btnQ3_2_2Click(Sender: TObject);  
begin  
    // Question 3.2.2  
    if objBeehive.checkHealthStatus then  
        lblQ3_2_2.Caption := 'Beehive is ready to harvest.'  
    else  
        lblQ3_2_2.Caption := 'Beehive is not ready to harvest.';  
    btnQ3_2_3.Enabled := objBeehive.checkHealthStatus;  
end;  
  
// =====  
// Question 3.2.3           4 marks  
// =====  
procedure TfrmQuestion3_2.btnQ3_2_3Click(Sender: TObject);  
var  
    rAmount: Real;  
begin  
    // Provided code  
    redQ3_2.Clear;  
  
    // Question 3.2.3  
    rAmount := StrToFloat(edtQ3_2_3.Text);  
    objBeehive.updateBeehiveDetails(rAmount);  
    redQ3_2.Lines.Add('Harvest number:' +  
        IntToStr(objBeehive.getNoOfHarvests));  
    redQ3_2.Lines.Add(objBeehive.toString);  
end;
```

```
// =====  
// Question 3.2.4          3 marks  
// =====  
procedure TfrmQuestion3_2.btnQ3_2_4Click(Sender: TObject);  
begin  
    // Question 3.2.4  
    ShowMessage('Average honey harvested = ' + FloatToStrF  
        (objBeehive.calcAverage, ffFixed, 6, 2));  
end;  
  
// =====  
// Question 3.2.5          2 marks  
// =====  
procedure TfrmQuestion3_2.btnQ3_2_5Click(Sender: TObject);  
begin  
    objBeehive.setPests;  
    btnQ3_2_3.Enabled := false;  
end;  
  
// =====  
// Provided code  
// =====  
  
{ $REGION 'Provided code - DO NOT MODIFY' }  
  
procedure TfrmQuestion3_2.FormCreate(Sender: TObject);  
begin  
    redQ3_2.clear;  
    redQ3_2.Paragraph.tabcount := 1;  
    redQ3_2.Paragraph.tab[0] := 150;  
  
    btnQ3_2_3.Enabled := False;  
end;  
{ $ENDREGION }  
  
end.
```


ANNEXURE H: SOLUTION FOR QUESTION 4

```
unit Question4_U;

interface

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

type
  TfrmQuestion4 = class(TForm)
    gbxQ4_1: TGroupBox;
    gbxQ4_2: TGroupBox;
    btnQ4_1_1: TButton;
    btnQ4_1_2: TButton;
    btnQ4_2: TButton;
    memQ4_2: TMemo;
    redQ4_1: TRichEdit;
    procedure btnQ4_1_1Click(Sender: TObject);
    procedure btnQ4_1_2Click(Sender: TObject);
    procedure btnQ4_2Click(Sender: TObject);

  private
    { Private declarations }
  public
    procedure Display;
  end;

var
  frmQuestion4: TfrmQuestion4;
  // Provided code
  arrSoil: array [1 .. 10] of String = (
    '20:50:10',
    '40:30:30',
    '30:28:30',
    '60:20:20',
    '25:30:45',
    '50:40:10',
    '30:55:15',
    '45:35:20',
    '35:0:55',
    '55:25:20'
  );
  arr2DSoil: array [1 .. 10, 1 .. 3] of Real;

implementation

{$R *.dfm}
```

```
// =====
// Question 4.1.1           8 marks
// =====
procedure TfrmQuestion4.btnQ4_1_1Click(Sender: TObject);
var
    iRow, iCol, iPos: Integer;
begin
    // Question 4.1.1

    for iRow := 1 to 10 do
    begin
        for iCol := 1 to 2 do
        begin
            iPos := pos(':', arrSoil[iRow]);
            arr2DSoil[iRow, iCol] := StrToInt(copy(arrSoil[iRow], 1, iPos - 1));
            delete(arrSoil[iRow], 1, iPos);
        end;
        arr2DSoil[iRow, iCol] := StrToInt(arrSoil[iRow]);
    end;

    // Provided code
    redQ4_1.Clear;
    Display;
end;

// =====
// Question 4.1.2           9 marks
// =====
procedure TfrmQuestion4.btnQ4_1_2Click(Sender: TObject);
var
    iRow, iCol, A: Integer;
    rClay, rSand, rSilt, rSum: Real;
begin
    // Provided code
    redQ4_1.Clear;
    redQ4_1.lines.add('Hectares adjusted:');
    // Question 4.1.2
    for iRow := 1 to 10 do
    begin
        rSum := 0;
        for iCol := 1 to 3 do
            rSum := rSum + arr2DSoil[iRow, iCol];

        if (rSum < 100) then
        begin
            for A := 1 to 3 do
                arr2DSoil[iRow, A] := (arr2DSoil[iRow, A] / rSum * 100);
            redQ4_1.Lines.Add('Hectare ' + IntToStr(iRow));
        end;
    end;

    // Provided code
    redQ4_1.Lines.Add(#13 + 'Updated data: ');
    redQ4_1.Lines.Add('=====');
    Display;
end;

// =====
Copyright reserved
```

// Question 4.2

13 marks

```
// =====
procedure TfrmQuestion4.btnQ4_2Click(Sender: TObject);
var
    rTFactor, rTNum: Real;
    iValue: Integer;
    iLoop1, iLoop2, iLoop3: Integer;
begin
    // Provided code
    memQ4_2.Clear;
    memQ4_2.Lines.Add('Nutrient markers:');
    memQ4_2.Lines.Add('=====');

    // Question 4.2
    For iLoop1 := 1 to 50000 do
    begin
        rTNum := 0;
        For iLoop2 := 1 to Length(IntToStr(iLoop1)) do
        begin
            rTFactor := 1;
            For iLoop3 := 1 to StrToInt(IntToStr(iLoop1)[iLoop2]) do
                rTFactor := rTFactor * iLoop3;
                rTNum := rTNum + rTFactor;
            end;
            if rTNum = iLoop1 then
                memQ4_2.Lines.Add(IntToStr(iLoop1));
            end;
        end;
    end;

//=====
// Display Method
// =====
// Provided code
procedure TfrmQuestion4.Display;
var
    iRow, iCol: Integer;
    sTotal: String;
begin
    redQ4_1.Lines.Add('Hectare' + #9 + 'Clay' + #9 + 'Sand' + #9 + 'Silt');
    For iRow := 1 to 10 do
    begin
        sTotal := IntToStr(iRow) + #9#9;
        For iCol := 1 to 2 do
            sTotal := sTotal + FloatToStrF(arr2DSoil[iRow, iCol],
                ffFixed, 8, 1) + #9;
        sTotal := sTotal + FloatToStrF(arr2DSoil[iRow, 3], ffFixed, 8, 1);
        redQ4_1.Lines.Add(sTotal);
    end;
end;

end.
```