



# basic education

Department:  
Basic Education  
**REPUBLIC OF SOUTH AFRICA**

## **SENIOR CERTIFICATE EXAMINATIONS/ NATIONAL SENIOR CERTIFICATE EXAMINATIONS**

**INFORMATION TECHNOLOGY P1**

**MAY/JUNE 2024**

**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 learners' 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 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 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	<b>Button [1.1 – Gender]</b>	3	
	Set the caption to "Gender" ✓ Add the option "Male" ✓ Set the Columns property to 2 ✓		
1.2	<b>Button [1.2 – Random day]</b>	13	
	Declare an integer variable for the random number ✓ Generate random number in correct range (1 – 7) ✓ Display "Day:" ✓ and the random number converted to string ✓ in the label lblQ1_2 ✓ Set the list box item index ✓ to the random number ✓ Test if ✓ (random number = 1) ✓ or (random number = 7) ✓ Display 'Weekend' in edtQ1_2 ✓ Else ✓ Display 'Weekday' in edtQ1_2 ✓  <b>Alternative:</b> case lstQ1_2.ItemIndex of (2) 1, 7 : edtQ1_2.Text := 'Weekend'; (2) 2..6 : edtQ1_2.Text := 'Weekday'; (2) end;		
1.3	<b>Button [1.3 – Calculate]</b>	11	
	Declare a suitable variable/s ✓ Extract years from the spin edit ✓ <i>Calculate bonus:</i> rBonus := Power (P,iYears) ✓ * SQR (SQR(P) ✓ / 7 * 20 ✓)  Test if checkbox manager is ticked ✓ rBonus := rBonus * 1.1 ✓ OR   rBonus := rBonus + rBonus * 0.1  Display bonus in a dialogue box ✓ formatted to currency and 2 decimal places ✓		

1.4	<p><b>Button [1.4 – Title case]</b></p> <p><b><i>Solution 1: Use of the for.. loop</i></b></p> <p>Add a space to the end of the sentence ✓          Initialise word string ✓          Loop ✓ from 1 to length of sSentence ✓              Check if a character in sSentence ✓ is &lt;&gt; to a space ✓                  Concatenate word string ✓ with character in sSentence ✓          Else              Convert first letter ✓ of word to uppercase ✓              Display word in redQ1_4 ✓              Clear ✓ word string ✓</p> <p><b><i>Alternatives:</i></b>  <b><i>Solution 2: Use of the while.. loop</i></b>  <b><i>Solution 3: Use of the repeat..until loop</i></b>  <b><i>See examples code in the code section</i></b></p> <p><b>Concepts:</b>          Mechanism to include last word (1)          Initialise word/ Counter/ Temp (1)</p> <p>Loop (1) with correct lower, upper limit/condition (1)</p> <p>Separate each word in two possible ways (4 marks):  <i>Method 1 (concatenate characters):</i>              Test if character (1) &lt;&gt; space (1)              Join character (1) to word (1)</p> <p><i>Method 2 (copy from sentence):</i>              Test if character (1) = space (1)              Copy word (1) from index 1 to space (1)</p> <p>Convert the first character (1) of the word to uppercase (1)          Display each word in redQ1_4 (1)</p> <p>Delete word (1) from index 1 to space (1)              OR          Clear (1) word (1)</p>	13	
	<b>TOTAL SECTION A:</b>	<b>40</b>	

**ANNEXURE B****QUESTION 2: MARKING GRID – SQL AND DATABASE PROGRAMMING**

<b>CENTRE NUMBER:</b>		<b>EXAMINATION NUMBER:</b>	
<b>QUESTION</b>	<b>DESCRIPTION</b>	<b>MAX. MARKS</b>	<b>LEARNER'S MARKS</b>
2.1	<b>SQL statements</b>		
2.1.1	<b>Button [2.1.1 – Free videos]</b>	3	
	SELECT Title, Duration, UploadDate, CreatorID ✓ FROM tblVideos ✓ WHERE FreeVideo = True ✓  <b>Also ACCEPT:</b> WHERE FreeVideo WHERE FreeVideo = Yes WHERE FreeVideo = -1		
2.1.2	<b>Button [2.1.2 – Check domain]</b>	5	
	SELECT CreatorName, Email, Country FROM tblCreators ✓ WHERE NOT Email ✓ LIKE "%@gmail%" ✓ AND ✓ Country = "South Africa" ✓  <b>Also ACCEPT:</b> Where Email NOT LIKE "%@gmail.com"		
2.1.3	<b>Button [2.1.3 – Latest videos]</b>	4	
	SELECT Top 3 ✓ UploadDate, VideoID, Title FROM tblVideos ✓ ORDER BY UploadDate ✓ DESC ✓		
2.1.4	<b>Button [2.1.4 – Videos per creator]</b>	8	
	SELECT CreatorID, ✓ Count(*) ✓ AS NumberUploaded ✓ FROM tblVideos ✓ GROUP BY ✓ CreatorID ✓ HAVING ✓ Count(*) > 5 ✓  <b>Also ACCEPT:</b> Count(field name)		
2.1.5	<b>Button [2.1.5 – Add new creator]</b>	4	
	INSERT INTO ✓ tblCreators ✓ VALUES ✓ ("C011", "TRISHKALOM", "trish@rsmarketing.co.za", "South Africa") ✓		
	<b>Subtotal:</b>	<b>24</b>	

**QUESTION 2: MARKING GRID (CONT.)**

2.2	<b>Database Manipulation</b>		
2.2.1	<b>Button [2.2.1 – Remove creator]</b>  Go to the first record in the tblCreators ✓ Use a loop to step through tblCreators ✓ Test if tblCreators['CreatorName'] = sCreatorName ✓ Go to the first record in the tblVideos ✓ Use a loop to step through tblVideos ✓  Test if (tblCreators ['CreatorID'] = tblVideos ['CreatorID']) ✓ tblVideos.Delete ✓ else ✓ tblVideos.Next ✓  End loop (tblVideos) tblCreators.Delete ✓ // End if tblCreators.Next ✓ End loop (tblCreators)  <b>NOTE:</b> Loops don't have to be nested. Can first find the CreatorID and use the CreatorID to delete the video records from tblVideos and then remove creator from tblCreators.	12	
2.2.2	<b>Button [2.2.2 – Change upload date]</b>  tblVideos.Edit; ✓ tblVideos ['UploadDate'] ✓ := Date ✓ tblVideos.Post; ✓  <b>Also ACCEPT:</b> DateToStr(Date) DateToStr(DateOf(Now)) Formatdatetime('yyyy/mm/dd',Now())	4	
	<b>Subtotal:</b>	<b>16</b>	
	<b>TOTAL SECTION B:</b>	<b>40</b>	

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

<b>CENTRE NUMBER:</b>		<b>EXAMINATION NUMBER:</b>	
<b>QUESTION</b>	<b>DESCRIPTION</b>	<b>MAX. MARKS</b>	<b>LEARNER'S MARKS</b>
3.1.1	<b>Constructor method:</b>  Heading ✓ with two parameter values of type string ✓ Assign correct parameter values to fAlbumTitle and fArtist ✓ Assign FALSE to fHighRanking ✓ Assign 0 to fPoints ✓	<b>5</b>	
3.1.2	<b>getPoints function:</b>  Function heading with integer return type ✓ fPoints assigned to result ✓	<b>2</b>	
3.1.3	<b>updatePoints procedure:</b>  Procedure heading ✓ with three integer parameters ✓ fPoints = ✓ album sales * 100 ✓ + downloadSongs * 10 ✓ + streamSongs ✓	<b>6</b>	
3.1.4	<b>setRanking procedure:</b>  Procedure heading with integer parameter ✓ If iNumWeeks > 4 ✓ Set fHighRanking to true ✓	<b>3</b>	
3.1.5	<b>determineStatus function:</b>  Test if fHighRanking = true ✓ Test if fPoints >= 5000 AND ✓ fPoints < 10000 ✓ Assign Gold to status ✓ Test if (fPoints >= 10000) ✓ Assign Platinum to status ✓ Result = status ✓	<b>7</b>	
	<b>Subtotal: Object class</b>	<b>23</b>	





#### QUESTION 4: MARKING GRID – PROBLEM SOLVING

CENTRE NUMBER:		EXAMINATION NUMBER:	
QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
4.1	<p><b>Button [4.1 – Sort]</b></p> <p>Loop ✓ I from 1 to length(arrPosition) ✓              Correct nesting ✓              Loop J from 1 to length(arrPosition) – 1 ✓                  Test if arrPosition[J] &gt; ✓                      arrPosition[J + 1] ✓                      iTemp := arrPosition[J]; ✓                      arrPosition[J] := arrPosition[J + 1]; ✓                      arrPosition[J + 1] := iTemp; ✓                  swop arrSongs ✓ at correct index ✓              End J loop          End I loop</p> <p><b>Alternative:</b>          for A := 1 to length(arrPosition) - 1 do (2)              for B := A+1 to length(arrPosition) do (2)                  if (arrPosition[A] &gt; arrPosition[B]) then (2)                      begin                          iTemp := arrPosition[A]; (1)                          arrPosition[A] := arrPosition[B]; (1)                          arrPosition[B] := iTemp; (1)                          sTemp := arrSongs[A];                          arrSongs[A] := arrSongs[B];                          arrSongs[B] := sTemp;                      end;                  } (2)</p> <p>Also ACCEPT:          Instead of Length(arrPosition) use 20          Instead of Length(arrPosition) - 1 use 19</p>	11	

4.2	<p><b>Button [4.2 – New chart]</b></p> <p>Display the headings (Song, Position and Movement) in redQ4 ✓</p> <p><b>File handling:</b>  AssignFile(tFile, 'Top20.txt') ✓  Reset(tFile) ✓  Loop through the file / Loop I 1 to 20 ✓  Read song from the text file ✓</p> <p><b>Movement changes:</b>  Initialise counter J to 0 ✓  Initialise flag to false ✓</p> <p>Loop while J &lt; 20 and Flag = false ✓  Increment J ✓  if song from text file = arrSongs[J] ✓  set flag to true ✓  if J &gt; I ✓  sMovement = intToStr(J - I) + ' UP' ✓  else if J &lt; I ✓  sMovement = intToStr(I - J) + ' DOWN' ✓  else  sMovement = 'SAME POSITION' ✓  if song not found in the text file / flag = false ✓  sMovement = 'NEW' ✓  Display song from text file, position and sMovement in redQ4 ✓</p> <p><b>Concepts:</b>  <b>Display</b>  Display headings to redQ4 (1)  Display song, position and movement to redQ4 (1)  <b>File Handling</b>  AssignFile (1)  Reset (1)  Loop though file (1)  Read from file (1)  <b>Movement</b>  <i>Use counters/indexes to determine the song position (4 marks):</i>  Initialise counters (1)  Loop (1)  Increment J (1)  Test if the song in the textfile = song in arrSongs (1)  Determine movement - up (2)  Determine movement - down (2)  Determine movement - same position (1)  Use a flag (2) to determine new songs (2)</p>	19	
	TOTAL SECTION D:	30	

**SUMMARY OF LEARNER'S MARKS:**

CENTER 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**

```
//=====
// 1.1 - Gender                                     3 marks
// =====
```

```
procedure TfrmQuestion1.btn1_1Click(Sender: TObject);
begin
  rgpQ1_1.Caption:='Gender';
  rgpQ1_1.Items.Add('Male');
  rgpQ1_1.Columns:=2;
end;
```

```
//=====
// 1.2 - Random day                               13 marks
// =====
```

```
procedure TfrmQuestion1.btnQ1_2Click(Sender: TObject);
var
  iRand: integer;
begin
  iRand := RandomRange(1,8);
  lstQ1_2.ItemIndex:= iRand;
  lblQ1_2.Caption := 'Day: '+IntToStr(iRand);
  if iRand IN [1, 7] then
    begin
      edtQ1_2.Text:= 'Weekend';
    end
  else
    begin
      edtQ1_2.Text:= 'Weekday';
    end;
end;
```

```
//=====
// 1.3 - Calculate                                 11 marks
// =====
```

```
procedure TfrmQuestion1.btnQ1_3Click(Sender: TObject);
const
  P = 8;
var
  rBonus:Real;
  iYears:Integer;
begin
  iYears := spnQ1_3.Value;
  rBonus := Power(P,iYears) * SQRT(SQR(P) / 7 * 20);
  if chkQ1_3.Checked then
    begin
      rBonus:= rBonus * 1.1;
    end;
  ShowMessage(FloatToStrF(rBonus,ffCurrency,10,2));
end;
```

```
//=====
// 1.4 - Title case                                     13 marks
// =====
```

```
procedure TfrmQuestion1.btnQ1_4Click(Sender: TObject);
var K : integer ;
    sSentence, sWord, sTitleCase : String;
begin
    // Provided code
    redQ1_4.Clear;
    sSentence := InputBox('', 'Enter sentence', 'Unlock the power of
technology and ignite innovation');
    // sSentence := InputBox('', 'Enter sentence', 'Let innovation be your
guiding star as you navigate the realms of cyberspace');

    // 1.4 - Title case
    // for loop - Solution 1
    sSentence := sSentence + ' ';
    sWord := '';
    for K := 1 to Length(sSentence) do
        begin
            if sSentence[K] <> ' ' then
                sWord := sWord + sSentence[K]
            else
                begin
                    sWord[1] := Upcase(sWord[1]);
                    redQ1_4.Lines.Add(sWord);
                    sWord := '';
                end;
            end;
        end;

    { while loop - Solution 2
    while sSentence <> '' do
        begin
            iPosWord := Pos(' ', sSentence);
            if iPosWord > 0 then
                begin
                    sWord := Copy(sSentence, 1, iPosWord - 1);
                    sWord := UPPERCASE(sWord[1]) +
                        Copy(sWord, 2, Length(sWord));
                    sSentence := Copy(sSentence, iPosWord + 1,
                        Length(sSentence) - iPosWord);
                end
            else
                if iPosWord = 0 then
                    begin
                        sWord := UPPERCASE(sSentence[1]) +
                            Copy(sSentence, 2, Length(sSentence));
                        sSentence := '';
                    end;
                end;
            redQ1_4.Lines.Add(sWord);
        end; }
end;
```

```
{ repeat..until loop - Solution 3

  i := 0;
  repeat
    inc(i);
    if sSentence[i] = ' ' then
  begin
    sTemp := Copy(sSentence, 1, i);
    Delete(sSentence, 1, i);
    sTemp := UpperCase(sTemp[1]) + Copy(sTemp, 2, length(sTemp));
    redQ1_4.Lines.Add(sTemp);
    i := 0;
  end;
until sSentence = ' ';}

end.
```

**ANNEXURE F: SOLUTION FOR QUESTION 2**

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

// =====
// 2.1.1 - Free videos                                     3 marks
// =====

sSQL1 := 'SELECT Title, Duration, UploadDate, CreatorID ' +
        'FROM tblVideos ' +
        'WHERE FreeVideo = True';

// =====
// 2.1.2 - Check domain                                   5 marks
// =====

sSQL2 := 'SELECT CreatorName, Email, Country ' +
        'FROM tblCreators ' +
        'WHERE Email NOT LIKE "%@gmail.com" AND ' +
        'Country = "South Africa"';

// =====
// 2.1.3 - Latest videos                                   4 marks
// =====

sSQL3 := 'SELECT Top 3 UploadDate, VideoID, Title ' +
        'FROM tblVideos ' +
        'ORDER BY UploadDate DESC';

// =====
// 2.1.4 - Videos per creator                             8 marks
// =====

sSQL4 := 'SELECT CreatorID, ' +
        'Count(*) AS NumberUploaded ' +
        'FROM tblVideos ' +
        'GROUP BY CreatorID ' +
        'HAVING Count(*) > 5';

// =====
// 2.1.5 - Add new creator                                 4 marks
// =====

sSQL5 := 'INSERT INTO tblCreators (CreatorID, CreatorName,
        Email, Country) ' +
        'VALUES ("C011", "TRISHKALOM", "trish@rsmarketing.co.za",
        "South Africa")';
```

```
// =====  
// 2.2 - Section Delphi code  
// =====  
  
// =====  
// 2.2.1 - Remove creator 12 marks  
// =====  
  
procedure TfrmQuestion2.btnQ2_2_1Click(Sender: TObject);  
var  
    sCreatorName : String;  
begin  
    // 2.2.1 - Remove creator  
    sCreatorName := cmbQ2_2_1.Text;  
    tblCreators.First;  
    while NOT tblCreators.Eof do  
        begin  
            if tblCreators['CreatorName'] = sCreatorName then  
                begin  
                    tblVideos.First;  
                    while NOT tblVideos.Eof do  
                        begin  
                            if tblCreators['CreatorID'] = tblVideos['CreatorID'] then  
                                tblVideos.Delete  
                            else  
                                tblVideos.Next;  
                            end;  
                        end;  
                    tblCreators.Delete;  
                    end;  
                    tblCreators.Next;  
                end;  
            end;  
        end;  
  
    // Provided code  
    ShowMessage('Records deleted successfully');  
end;  
  
// =====  
// 2.2.2 - Change upload date 4 marks  
// =====  
  
procedure TfrmQuestion2.btnQ2_2_2Click(Sender: TObject);  
begin  
    tblVideos.Edit;  
    tblVideos['UploadDate'] := Date;  
    tblVideos.Post;  
end;
```



```
// =====  
// {$ENDREGION}  
// =====  
// {$REGION 'Provided code: Setup DB connections - DO NOT CHANGE!'}  
// =====  
  
procedure TfrmQuestion2.FormClose(Sender: TObject; var Action:  
TCloseAction);  
begin  
    // Disconnects from database and closes all open connections  
    dbCONN.dbDisconnect;  
end;  
  
procedure TfrmQuestion2.FormShow(Sender: TObject);  
begin  
    // Sets up the connection to database and opens the tables.  
    dbCONN := TConnection.Create;  
    dbCONN.dbConnect;  
    tblManufacturers := dbCONN.tblOne;  
    tblProducts := dbCONN.tblMany;  
    dbCONN.setupGrids(dbgManufacturers, dbgProducts, dbgSQL);  
    pgcDBAdmin.ActivePageIndex := 0;  
end;  
  
// =====  
// {$ENDREGION}  
// =====  
  
end.
```

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

```
unit Album_U;

interface

uses
    SysUtils, StdCtrls, Dialogs, Math;

type
    TAlbum = class(TObject)
    private
        fAlbumTitle: String;
        fArtist: String;
        fHighRanking: Boolean;
        fPoints: Integer;

    public
// Provided code
        function toString: String;
        function determineStatus: String;
        //
=====

        constructor Create(sAlbumTitle, sArtist: String);
        procedure updatePoints(iAlbumSales, iSongsDownload, iSongsStream:
                                Integer);
        procedure setRanking(iNumWeeks: Integer);
        function getPoints: Integer;
end;

implementation

// =====
// Provided code
// =====
function TAlbum.toString: String;
begin
    Result := 'Title: ' + fAlbumTitle + #13 + 'Artist: ' + fArtist + #13 +
'High ranking: ' +
        BoolToStr(fHighRanking, true) + #13 + 'Number of points: ' + IntToStr
        (fPoints);
end;
// =====
```

```
// =====
// 3.1.1 Constructor Create                                     5 marks
// =====

constructor TAlbum.Create(sAlbumTitle, sArtist: String);
begin
    fAlbumTitle := sAlbumTitle;
    fArtist := sArtist;
    fHighRanking := false;
    fPoints := 0;
end;

// =====
// 3.1.2 Function getPoints                                     2 marks
// =====

function TAlbum.getPoints: integer;
begin
    Result := fPoints;
end;

// =====
// 3.1.3 Procedure updatePoints                                 6 marks
// =====

procedure TAlbum.updatePoints(iAlbumSales, iSongsDownload,
    iSongsStream: Integer);
begin
    fPoints := iAlbumSales * 100 + iSongsDownload * 10 + iSongsStream;
end;

// =====
// 3.1.4 Procedure setRanking                                   4 marks
// =====

procedure TAlbum.setRanking(iNumWeeks: integer);
begin
    if iNumWeeks > 4 then
        fHighRanking := true;
end;
```

```
// =====  
// 3.1.5 Function determineStatus                                7 marks  
// =====  
function TAlbum.determineStatus: String;  
var  
    sStatus: String;  
begin  
    // Provided code  
    sStatus := 'None';  
  
    // 3.1.5  
    if (fHighRanking) then  
        if (fPoints >= 5000) AND (fPoints < 10000) then  
            sStatus := 'Gold';  
        if (fPoints >= 10000) then  
            sStatus := 'Platinum';  
    Result := sStatus;  
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;
    redQ3: TRichEdit;
    btnQ3_2_1: TButton;
    gbxQ3_2_2: TGroupBox;
    btnQ3_2_2: TButton;
    Panel1: TPanel;
    Panel2: TPanel;
    btnQ3_2_3: TButton;
    Image1: TImage;
    Label6: TLabel;
    edtQ3_2_1: TEdit;
    Label2: TLabel;
    spnQ3_2_1: TSpinEdit;
    chbQ3_2_1: TCheckBox;
    Label1: TLabel;
    sedQ3_2_2: TSpinEdit;
    procedure btnQ3_2_1Click(Sender: TObject);
    procedure btnQ3_2_2Click(Sender: TObject);
    procedure btnQ3_2_3Click(Sender: TObject);
  private
    const
      arrArtist: array [1 .. 3] of string = ('SZA', 'Morgan Wallen',
                                              'People');
  public

  end;

var
  frmQuestion3: TfrmQuestion3;
  objAlbum: TAlbum;
  iSold, iDownloaded, iStreamed : integer;
implementation

{$R *.dfm}
```

```
// =====
// 3.2.1 Instantiate album object                                5 marks
// =====

procedure TForm1.btnQ3_2_1Click(Sender: TObject);
var
    sAlbum, sArtist: String;
begin
    sAlbum := cmbQ3_2_1.Text;
    sArtist := edtQ3_2_1.Text;
    objAlbum := TAlbum.Create(sAlbum, sArtist);

    // Provided code
    ShowMessage('Album object has been instantiated successfully.');
```

end;

```
// =====
// 3.2.2 Calculate points                                        5 marks
// =====

procedure TfrmQuestion3.btnQ3_2_2Click(Sender: TObject);
begin
    objAlbum.updatePoints(iSold, iDownloaded, iStreamed);
    lblQ3_2_2.Caption := IntToStr(objAlbum.getPoints);
end;
```

```
// =====
// 3.2.3 Set ranking                                           4 marks
// =====

procedure TfrmQuestion3.btnQ3_2_3Click(Sender: TObject);
var
    iNumWeeks : integer;
begin
    iNumWeeks := StrToInt(InputBox('Number of weeks ranked 1', 'Enter number
of weeks', ''));
    objAlbum.setRanking(iNumWeeks);
end;
```

```
// =====
// 3.2.4 Display album details                                  3 marks
// =====

procedure TForm1.btnQ3_2_4Click(Sender: TObject);
begin
    redQ3.Clear;
    redQ3.Lines.Add(objAlbum.toString);
    redQ3.Lines.Add('Status of album: ' +
    objAlbum.determineStatus);end;
```

```
// Provided code - do not change
// =====

procedure TfrmQuestion3.cmbQ3_2_1Change(Sender: TObject);
begin
    edtQ3_2_1.Text := arrArtist[cmbQ3_2_1.ItemIndex + 1];
    iSold := Random(100);
    iDownloaded := Random(500);
    iStreamed := Random(500);

    edtSold.Text := IntToStr(iSold);
    edtDownloaded.Text := IntToStr(iDownloaded);
    edtStreamed.Text := IntToStr(iStreamed);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, math;
type
  TfrmQuestion4 = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    btnQ4_2: TButton;
    redQ4: TRichEdit;
    btnDisplay: TButton;
    GroupBox1: TGroupBox;
    btnQ4_1: TButton;
    Image1: TImage;
    GroupBox2: TGroupBox;
    GroupBox3: TGroupBox;
    procedure btnQ4_2Click(Sender: TObject);
    procedure btnDisplayClick(Sender: TObject);
    procedure btnQ4_1Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
  private
    procedure DisplayArrays;
    { Private declarations }
  public
    { Public declarations }
  end;
var
  frmQuestion4: TfrmQuestion4;

  arrSongs: array [1 .. 20] of String = (
    'Castle of Hope', 'Deep Green Hills', 'Backseat Kiss', 'Earning
    Nocturno', 'Edges of Dawing', 'Free Future', 'Heart Hymn', 'Heroic Flavor',
    'Me and You', 'New York Dirt', 'Not Night', 'Adagio', 'Running Study', 'So
    Hard Spring', 'Sound of Illusion', 'The Celebration', 'Unexpected Skies',
    'Wait for Friends', 'Warm Heart', 'Winter Friends');

  arrPosition: array [1 .. 20] of integer = (
    4, 6, 11, 20, 2, 12, 19, 5, 1, 10, 14, 13, 17, 9, 3, 18, 16, 7, 8, 15);

implementation

{$R *.dfm}
```



```
// =====  
// 4.1 - Sort 11 marks  
// =====
```

```
procedure TfrmQuestion4.btnQ4_1Click(Sender: TObject);
```

```
var
```

```
    I, iTemp: integer;
```

```
    J: integer;
```

```
    sTemp: String;
```

```
begin
```

```
    // Provided code
```

```
    redQ4.Clear;
```

```
    // Question 4.1
```

```
    for I := 1 to length(arrPosition) do
```

```
    begin
```

```
        for J := 1 to length(arrPosition) - 1 do
```

```
        begin
```

```
            if arrPosition[J] > arrPosition[J + 1] then
```

```
            begin
```

```
                iTemp := arrPosition[J];
```

```
                arrPosition[J] := arrPosition[J + 1];
```

```
                arrPosition[J + 1] := iTemp;
```

```
                sTemp := arrSongs[J];
```

```
                arrSongs[J] := arrSongs[J + 1];
```

```
                arrSongs[J + 1] := sTemp;
```

```
            end;
```

```
        end;
```

```
    end;
```

```
    DisplayArrays;
```

```
end;
```

```
// =====  
// 4.2 New chart 19 marks  
// =====
```

```
procedure TfrmQuestion4.btnQ4_2Click(Sender: TObject);  
var  
    tFile: TextFile;  
    J, iNew: integer;  
    bFound: boolean;  
    sNewSong, sMsg: String;  
  
    // Variables for Alternative 2 and 3  
    // tFile: TextFile;  
    // arrNewSongs: array [1 .. 20] of String;  
    // I: integer;  
    // J, iDiff, iPos: integer;  
    // bFlag, bFound: boolean;  
    // sLine, sOutput: String;  
begin  
    // Question 4.2  
  
    redQ4.Clear;  
    AssignFile(tFile, 'Top20.txt');  
    Reset(tFile);  
  
    redQ4.lines.add('Song' + #9 + 'Position' + #9 + 'Movement');  
  
    for iNew := 1 to 20 do  
    begin  
        readln(tFile, sNewSong);  
        J := 0;  
        bFound := false;  
        while (J < 20) AND (NOT bFound) do  
        begin  
            inc(J);  
            if sNewSong = arrSongs[J] then  
            begin  
                bFound := true;  
                if J > iNew then  
                    sMsg := IntToStr(J - iNew) + ' UP'  
                else if iNew > J then  
                    sMsg := IntToStr(iNew - J) + ' DOWN'  
                else  
                    sMsg := 'SAME POSITION';  
            end;  
        end;  
        if NOT bFound then  
            sMsg := 'NEW';  
  
        redQ4.lines.add(sNewSong + #9 + IntToStr(iNew) + #9 + sMsg);  
  
    end;  
end;
```

```
{ //Alternative 2
redQ4.Clear;

AssignFile(tFile, 'Top20.txt');

try
    reset(tFile);

    for I := 1 to 20 do
        begin

            readln(tFile, arrNewSongs[I]);

        end;
        redQ4.lines.add('Song' + #9 + 'Position' + #9 + 'Movement');

    for I := 1 to length(arrNewSongs) do
        begin

            J := 0;
            bFlag := true;
            while (J < 20) AND (bFlag) do
                begin
                    inc(J);
                    if arrNewSongs[I] = arrSongs[J] then
                        begin
                            if J - I > 0 then
                                begin
                                    sLine := arrNewSongs[I] + #9 + intToStr(I) + #9 + '(' +
                                        intToStr(abs(J - I)) + ' UP)';
                                end
                            else if J - I < 0 then
                                Begin
                                    sLine := arrNewSongs[I] + #9 + intToStr(I) + #9
                                        + '(' + intToStr(abs(J - I)) + ' DOWN)';
                                End
                            else
                                begin
                                    sLine := arrNewSongs[I] + #9 + intToStr(I) + #9
                                        + '(SAME POSITION)';
                                end;
                                bFlag := false;
                            end
                        else
                            begin
                                sLine := arrNewSongs[I] + #9 + intToStr(I) + #9 + '(NEW)';
                            end;
                        end;
                    redQ4.lines.add(sLine);
                end;
            except
                ShowMessage('File not found');
                Application.Terminate;
            end;
        }
}
```

```
{ //Alternative 3
AssignFile(tFile, 'Top20.txt');
Reset(tFile);
j:=0;
while not eof(tFile) do
Begin
  Readln(tFile, sLine);
  bFlag:= False;
  Inc(j);
  i:=0;
  sOutput:= sLine+#9+IntToStr(j);
  while(bFlag = false) AND (i<20) do
  begin
    Inc(i);
    if arrSongs[i] = sLine then
    begin
      bFlag := True;

      iDiff := i - j;
      if iDiff < 0 then
      begin
        sOutput := sOutput+ #9+'('+IntToStr(ABS(iDiff))+ ' DOWN)';
      end
      else if iDiff > 0 then
      begin
        sOutput := sOutput+ #9+'('+IntToStr(iDiff)+ ' UP)';
      end
      else if iDiff = 0 then
      begin
        sOutput := sOutput+ #9+'SAME POSITION';
      end ;

    end; //if bflag - true
  end; //while for array

  if bFlag = False then
  begin
    sOutput:= sOutput+#9+'NEW';
  end;

  redQ4.Lines.Add(sOutput);
End; //while for file
CloseFile(tFile);
end; }
```

```
// =====  
// Provided code  
// =====  
procedure TfrmQuestion4.FormShow(Sender: TObject);  
begin  
    redQ4.Paragraph.TabCount := 3;  
    redQ4.Paragraph.Tab[0] := 0;  
    redQ4.Paragraph.Tab[1] := 120;  
    redQ4.Paragraph.Tab[2] := 180;  
end;  
  
procedure TfrmQuestion4.DisplayArrays;  
var  
    I: Integer;  
begin  
    // Provided code  
    redQ4.lines.add('TOP CHARTS');  
    redQ4.lines.add(format('%-20s%-15s%-5s', ['Songs', 'Artist',  
'Position']));  
    for I := 1 to length(arrSongs) do  
        redQ4.lines.add(format('%-20s%-15s%-5d', [arrSongs[I], arrArtists[I],  
arrPosition[I]]));  
    end;  
  
end.  
// =====  
End of provided code  
// =====
```