basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

SENIOR CERTIFICATE EXAMINATIONS/ NATIONAL SENIOR CERTIFICATE EXAMINATIONS

INFORMATION TECHNOLOGY P1

2022

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 to 10) include the marking grid for each question.

- **Annexures E, F, G** and **H** (pages 11 to 25) contain examples of solutions for Questions 1 to 4 in programming code.

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

ANNEXURE A

QUESTION 1: MARKING GRID – GENERAL PROGRAMMING SKILLS

CENTRE NUMBER:		EXAMINATION NUMBER:		

QUESTION	DESCRIPTION		MAX. MARKS	LEARNER'S MARKS
1.1	**Button [1.1 – Add device]** Add the 'Tablet' item to rgpQ1_1 ✓ Change colour of rgpQ1_1 to cream ✓ Set item index to 0 ✓		3	
1.2	**Button [1.2 – Apply]**			
	1.2.1	Retrieve age from **edtQ1_2** ✓ and convert to integer ✓	2	
	1.2.2	If (checkbox not ticked) ✓ then Display suitable message using ShowMessageDialog ✓	2	
	1.2.3	if (age < 16) ✓ YearToApply = CURRENT_YEAR + (16 – iAge) ✓ Display suitable message using ShowMessageDialog in first line of text ✓ Showing year to apply in next line of text ✓ converted value from int to string ✓	5	
	1.2.4	if (checkbox is checked and age >= 16) ✓ Change the text of **btnQ1_2** to 'SUCCESSFUL' ✓ Note: Also accept **Else** with correct nested with both conditions	2	

| 1.3 | **Button [1.3 – Fractions]**

 Initialise all three variables ✓
 Total = 0
 Top = 1
 Bottom = 1 // check that division by 0 is not possible – loose <br> mark at increment
 While loop ✓ condition: sum of terms <= 4 ✓
    Term = Top ✓ / bottom ✓
    Increment bottom ✓
    Add term to Total ✓
 Display total in **redQ1_3** converted to string ✓
    and 4 decimal places ✓
 Display number of terms in **redQ1_3** converted to string ✓

 Alternative to while loop:
 Repeat (1)
    Total := Total (1) + (Top (1) / Bottom); (1)
    Inc(Bottom); (1)
 Until Total > 4; (1) | **10** | |

1.4.1	**Button [1.4.1 – Count letter]**
	
	Extract letter from **edtQ1_4_1** ✓ and convert sentence and letter to either uppercase/lowercase ✓
	
	Initialise counter variable to 0 ✓
	
	Loop from 1~~0~~ to Length of sentence ✓
	   Check if character in sentence ✓ = letter ✓
	     Increment counter ✓
	Display the count on the panel ✓
	
	Also accept alternative solutions.
		8	

1.4.2	**Button [1.4.2 – Longest word]**
	
	Add space at the end of sentence ✓
	Initialise iLarge = 0 ✓ // any value < 1
	
	Loop while position of space in sentence > 0 ✓
	   Get position of space and calculate length of word ✓
	     Test if length of word > iLarge ✓
	       Store length of this word in iLarge ✓
	   Delete characters in sentence up to space ✓
	
	Display message indicating the length of the longest word ✓
	
	Concepts:
	
	Initialise variable for Longest to 1 or less //(1)
	Loop through sentence //(1)
	  Identify individual words //(1) using spaces //(1)
	  Determine length of word //(1)
	  Test if length of word > Longest //(1)
	    Set Longest to length of word //(1)
	Display the Longest //(1)
		8	

	TOTAL SECTION A:	**40**	

ANNEXURE B

QUESTION 2: MARKING GRID – SQL AND DATABASE

CENTRE NUMBER:		EXAMINATION NUMBER:		
QUESTION	**DESCRIPTION**		**MAX. MARKS**	**LEARNER'S MARKS**
2.1	**SQL statements**			
2.1.1	**Button [2.1.1 – List of bookings]** SELECT BookingNum, BookingDate ✓ FROM tblBookings ✓ ORDER BY BookingDate ✓		**3**	
2.1.2	**Button [2.1.2 – Update test venue]** UPDATE tblBookings ✓ SET ✓ TestVenue = "Headquarters" ✓ WHERE TestVenue IS NULL ✓		**4**	
2.1.3	**Button [2.1.3 – Bellville bookings]** SELECT BookingNum,LDriverName,LDriverSurname ✓ FROM tblBookings,tblLDrivers ✓ WHERE tblBookings.LDriverID = tblLDrivers.LDriverID ✓ AND tblBookings.TestVenue = "Bellville" ✓// or no table name // also accept aliases		**4**	
2.1.4	**Button [2.1.4 – Licence types]** SELECT LEFT(BookingNum,1) ✓ AS [LicenceTypes], ✓ count(BookingNum) ✓ AS [Number] FROM tblBookings GROUP BY ✓ LEFT(BookingNum,1) ✓ Note: Count can include any field from the tblBookings(including *)		**5**	

| 2.1.5 | **Button [2.1.5 – Remove bookings]**

DELETE FROM tblBookings ✓
WHERE TestVenue = "Pretoria" ✓
AND ✓ BookingDate BETWEEN ✓
#2022/05/18# AND #2022/05/25# ✓

Alternative:
DELETE * FROM tblBookings   (1)
WHERE TestVenue = "Pretoria"   (1)
AND   (1)
(Year(BookingDate) = 2002
AND Month(BookingDate) = 5)
AND (Day(BookingDate) >=18   (1)
          AND Day(BookingDate) <=25)) (1) | **5** | |
| | **Subtotal:** | **21** | |

QUESTION 2: MARKING GRID (CONT.)

2.2	**DATABASE MANIPULATION using Delphi code**		
2.2.1	**Button - [2.2.1 – Bookings per gender]** Go to first record in tblBookings✓ Loop until the end of the tblBookings table ✓ Test if 7th digit ✓ is less than 5 ✓ Increase the female variable ✓ Else ✓ Increase the male variable ✓ Move to the next record ✓ Display the number of males and females	**8**	
2.2.2	**Button - [2.2.2 – Display bookings for a learner driver]** Go to the first record of tblBookings ✓ Loop until the end of the tblBookings table ✓ Test if the ID ✓ is same as input ID ✓ Display booking number ✓ and date (DateToStr) ✓ Move to the next record ✓	**7**	

| 2.2.3 | **Button - [2.2.3 – Add learner driver]**

tblLDrivers.Insert ✓

    tblLDrivers['LDriverID'] := '0405060708091'
    tblLDrivers['LDriverName'] := 'Trish'
    tblLDrivers['LDriverSurname'] := 'Malope'
    tblLDrivers['LDriverCell'] := '0710810911'

 tblLDrivers.Post ✓

✓✓ - mark allocation
1 mark for any one field assigned correctly
1 mark for correctly assigning all other fields | **4** | |
| | **Subtotal:** | **19** | |

| | **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	**Function compileTripNum:** Repeat ✓ Generate a random number ✓ of ~~four /~~ three digits ✓ Until the last digit is NOT zero ✓ Alternative: sTripNum = IntToStr(Random(10)) + // (1) IntToStr(Random(10)) + // (1) IntToStr(Random(9) + 1) // (1),last character is not a 0 // (1) Also accept other alternatives	**4**	
3.1.2	**Constructor Create:** Header with correct parameters✓ of the correct type ✓ Assign fDeparture, fDestination and fLoad to correct parameters ✓ fTripNumber = compileTripNum ✓ fDistance = 0 ✓	**5**	
3.1.3	**Function getDistance:** Function heading with correct Integer return type ✓ Result = fDistance ✓	**2**	
3.1.4	**Procedure setDistance:** Procedure heading with correct parameter ✓ fDistance = parameter value ✓	**2**	

| 3.1.5 | **Function determineTruckType:**

Function heading with correct string return type✓

Test if distance =0, ✓set result to 'No distance' ✓
Test if (fLoad <= 1000) ✓ then
     sTruck = 'Light truck' ✓
   else if fLoad <= 5000 then ✓
      sTruck = 'Medium truck'✓
       else ✓ sTruck = 'Heavy truck' ✓
Result = sTruck ✓

Alternative:
Function heading with correct string return type   // (1)

`if fLoad <= 1000 then //(1)`

`    sTruck = 'Light Truck' //(1)`

`if (fLoad > 1000)`

`        AND (fLoad <= 5000) then  //(1)`

`    sTruck = 'Medium Truck' //(1)`

`if fLoad > 5000 then  //(1)`

`    sTruck = 'Heavy Truck' //(1)`

`result = sTruck (1)` | **8** | |
| | **Subtotal:** | **21** | |

QUESTION 3: MARKING GRID (CONT.)

3.2.1	**Button [3.2.1 – Create trip]** Extract values from components ✓ Instantiate the object with the provided values objTrip:= TDeliveryTrip.create ✓ (sDepartureCity ,sDestinationCity, rLoad) Correct number of parameters ✓ Correct order of parameters ✓ Display the object details in the rich edit using the toString method ✓	**5**	
3.2.2	**Button [3.2.2 – Determine and set distance]** Assign and reset file ✓ Loop through the text file ✓ Read line from text file ✓ Extract the departure city from the text file ✓ Extract the destination city from the text file ✓ Test if the departure ✓ and destination cities ✓ of the text file is the same as the departure and destination cities in the object Determine the position of the delimiter # ✓ Extract the distance from the text file ✓ Call the setDistance method with distance as the argument ✓ Display the distance between the cities in the edit box **edtQ3_2_2** ✓ Use the toString method to show the updated information in the rich edit ✓	**12**	
3.2.3	**Button [3.2.3 – Determine truck type]** Call the determineTruckType method ✓ Display the truck type in the edit box **edtQ3_2_3** ✓	**2**	
	Subtotal:	**19**	

	TOTAL SECTION C:	**40**	

ANNEXURE D

QUESTION 4: MARKING GRID – PROBLEM-SOLVING PROGRAMMING

CENTRE NUMBER:		EXAMINATION NUMBER:	

SECTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS

4.1	**Button [4.1 - Display]**
	Display Heading
	Loop ✓from 1 to number of elements ✓
	   Display loop counter,✓ converted to string ✓,
	arrRegNumbers[k] ✓and arrEntryTime [k] ✓
	   In neat columns ✓ (e.g. #9)
		7	

4.2	**Button [4.2 – Exit parking area]**
	
	Extract regNum from cmbQ4 ✓
	Extract exit time from edtQ4 ✓
	
	Extract time from arrEntryTime
	   Loop 1 to number of elements in array ✓
	    If arrRegNumbers[k] = regnum ✓
	    Get inTime from arrEntryTime ✓
	    Get index for specific registration number ✓
	
	Determine time spent in parking area
	Extract and convert HourIn to integer ✓
	HourInMin = HourIn * 60 ✓
	Extract and convert MinIn to integer ✓
	MinutesIn = HourInMin + MinIn ✓
	
	Extract and convert HourOut to integer ⎫
	HourOutMin = HourOut * 60 ⎬ ✓
	Extract and convert MinOut to integer ⎭
	MinutesOut = HourOutMin + MinOut
	
	iTimeSpent = MinutesOut – MinutesIn ✓
	
	if iTimeSpent >= 0✓ then //(valid)
	
	Determine tariff
	  Case / if 0..30 : Tariff = 0 ⎫
	  Case / if 31 .. 120 : Tariff = 50 ⎬ ✓✓
	  Case / if 121 .. 240 : Tariff = 40 ⎭
	  Case / if > 240 : Tariff = 30		

	Determine total
	   Cost = Tariff * Ceil(iTimeSpent / 60); ✓
	
	Move elements up in arrays
	   Loop from index to length of arrRegNumbers -1 ✓
	      Move element up in arrRegNumbers ✓
	      Move element up in arrEntryTime ✓
	   Decrease counter
	
	Output
	   Display RegNumber, Entry time and Exit time
	   Display hours (iTimeSpent div 60)
	      and minutes (iTimeSpent mod 60) ✓✓
	   Display tariff in currency
	   Display total cost
	
	Else
	   Display Invalid Exit time ✓
		23	

	TOTAL SECTION D: **GRAND TOTAL:**	**30** **150**	

SUMMARY OF LEARNER'S MARKS:

	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
    rgpQ1_1.Items.Add('Tablet');
    rgpQ1_1.ItemIndex := 0;
    rgpQ1_1.Color := clCream;
end;
// ======================================================================
//                        Question 1.2 – 11 marks
// ======================================================================
procedure TfrmQuestion1.btnQ1_2Click(Sender: TObject);
const
  CURRENT_YEAR = 2022; // provided code
var
  iAge, iAppYear: integer;
begin
  iAge := StrToInt(edtQ1_2.Text);
  if (NOT(ckbSACitizen.checked)) then
      ShowMessage('The applicant must be a South African citizen.')
  else
  if iAge < 16 then
  begin
    iAppYear := CURRENT_YEAR + (16 - iAge);
    ShowMessage('The applicant is too young.' +  #13 +
                'Can apply in the year ' + IntToStr(iAppYear));
  end
  else
      btnQ1_2.Caption := 'SUCCESSFUL';
end;
// ======================================================================
//                        Question 1.3 – 10 marks
// ======================================================================
procedure TfrmQuestion1.btnQ1_3Click(Sender: TObject);
var
  iBottom, iTop : integer;
  rTerm, rTotal : real;
begin
  iBottom := 0;
  iTop := 1;
  rTotal := 0;
  while rTotal <= 4 do
     begin
     inc(iBottom);
     rTerm := iTop / iBottom;
     rTotal := rTotal + rTerm;
     end;
  redQ1_3.Lines.Add('Amount of terms: ' + IntToStr(iBottom));
  redQ1_3.Lines.Add('Total: ' + FloatToStrF(rTotal, ffFixed, 10, 4));
end;
```

```
// =====================================================================
                     Question 1.4.1 – 8 marks
// =====================================================================
procedure TfrmQuestion1.btn1_4_1Click(Sender: TObject);
var
  sSentence : String ;
  iCount, k  : integer ;
  cLetter : char;
begin
  sSentence := UpperCase(edtQ1_4.Text);
  cLetter := Upcase(edtQ1_4_1.Text[1]);
  iCount := 0;
  for k := 0 to Length(sSentence) do
  begin
      if (sSentence[k] = cLetter) then
         iCount := iCount + 1;
  end;
  pnlQ1_4_1.Caption := 'Number: ' + IntToStr(iCount);
end;


// =====================================================================
                     Question 1.4.2 – 8 marks
// =====================================================================
procedure TfrmQuestion1.btnQ1_4_2Click(Sender: TObject);
var
  sSentence : String ;
  iLarge, k, iLength : integer ;
begin
  sSentence := edtQ1_4.Text + ' ';
  iLarge := 0;
  while pos(' ', sSentence) > 0 do
   begin
     iLength := pos(' ',sSentence)- 1;
     if iLength > iLarge then
       iLarge := iLength;

     Delete(sSentence,1,iLength + 1);
   end;
   pnlQ1_4_2.Caption := 'Length of longest word: ' + IntToStr(iLarge);

 { //Alternative
  sSentence := edtQ1_4.Text + ' ';

  sLargest := '';
  iStart := 1;

  for k := 1 to Length(sSentence) do
    begin
      if sSentence[k] = ' ' then
        begin
          iEnd := k;
          sWord := Copy(sSentence, iStart, iEnd - iStart);
          if iLarge < Length(sWord) then
              iLarge := Length(sWord);
          iStart := iEnd + 1 ;
```

Please turn over

```
        end;
    end;
  redQ1_4_2.Lines.Add('Length of longest word: ' + IntToStr(iLarge));
end;     }
end.
```


ANNEXURE F: SOLUTION FOR QUESTION 2

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

  // --- Global variables provided ---
  tblLDrivers, tblBookings: TADOTable;
  qryDB: TADOQuery;

implementation

{$R *.dfm}
{$R+}

// Question 2.1 - SQL section

// ======================================================================
//                        Question 2.1.1 – 3 marks
// ======================================================================
procedure TfrmQuestion2.btnQ2_1_1Click(Sender: TObject);
var
  sSQL1: String;
begin
  sSQL1 := 'SELECT BookingNum, BookingDate FROM tblBookings ' +
  ' ORDER BY BookingDate';

  // Provided code - do not change
  dbCONN.RunSQL(sSQL1, dbgSQL);
end;

// ======================================================================
//                        Question 2.1.2 – 4 marks
// ======================================================================
procedure TfrmQuestion2.btnQ2_1_2Click(Sender: TObject);
var
  sSQL2: String;
  bChange: Boolean;
begin
  sSQL2 := 'UPDATE tblBookings SET TestVenue = "Headquarters" WHERE
           TestVenue IS NULL';

   // Provided code - do not change
  dbCONN.ExecuteSQL(sSQL2, bChange);
  if bChange then
    begin
      MessageDlg('Database updated', mtInformation, [mbOK], 0);
    end;
end;
```

      Please turn over

```
// =====================================================================
//                   Question 2.1.3 - 4 marks
// =====================================================================
procedure TfrmQuestion2.btnQ2_1_3Click(Sender: TObject);
var
  sSQL3: String;
begin
  sSQL3 := 'SELECT BookingNum,LDriverName,LDriverSurname FROM
           tblBookings,tblLDrivers WHERE tblBookings.LDriverID =
           tblLDrivers.LDriverID AND TestVenue = "Bellville"';

   // Provided code - do not change
  dbCONN.RunSQL(sSQL3,dbgSQL);
end;
// =====================================================================
//                   Question 2.1.4 - 5 marks
// =====================================================================
procedure TfrmQuestion2.btnQ2_1_4Click(Sender: TObject);
var
  sSQL4: String;
begin
  sSQL4 := 'SELECT LEFT(BookingNum,1) AS [LicenceTypes]
           Count(BookingNum) AS [Number] FROM tblBookings
           GROUP BY LEFT(BookingNum,1)';

   // Provided code - do not change
    dbCONN.RunSQL(sSQL4,dbgSQL);

end;


// =====================================================================
//                   Question 2.1.5 - 5 marks
// =====================================================================
procedure TfrmQuestion2.btnQ2_1_5Click(Sender: TObject);
var
  sSQL5: String;
  bChange: Boolean;
begin
  sSQL5 := DELETE FROM tblBookings WHERE TestVenue = "Pretoria" AND
           BookingDate BETWEEN #2022/05/18# AND #2022/05/25# ';

// Provided code - do not change
  if dbCONN.ExecuteSQL(sSQL5, bChange) then
    begin
      MessageDlg('Database updated', mtInformation, [mbOK], 0);
    end;
end;
```

Please turn over

```
// Question 2.2 - Delphi section

// =====================================================================
//                     Question 2.2.1 – 8 marks
// =====================================================================
procedure TfrmQuestion2.btnQ2_2_1Click(Sender: TObject);
var
  iMaleCount, iFemaleCount : Integer;
begin
//Provided code
  iMaleCount := 0;
  iFemaleCount := 0;

//Question 2.2.1
  tblBookings.First;
  while not tblBookings.Eof do
    begin
      if COPY(tblBookings['LDriverID'],7,1) IN ['0'.. '4'] then
          Inc(iFemaleCount)
      else
          Inc(iMaleCount);

      tblBookings.Next;
    end;

//Provided code
 redQ2.Lines.Add('Female: ' + IntToStr(iFemaleCount));
 redQ2.Lines.Add('Male: ' + IntToStr(iMaleCount));
end;

// =====================================================================
//                     Question 2.2.2 – 7 marks
// =====================================================================
procedure TfrmQuestion2.btnQ2_2_2Click(Sender: TObject);
var
  sID : String;
  bFound : Boolean;
  sOut : String;
begin
//Provided code
redQ2.clear;
tblLDrivers.First;
sID := InputBox('Enter learner driver`s ID','','0207280128342');

 //Question 2.2.2

tblBookings.First;
while not tblBookings.eof do
   begin
    if tblBookings['LDriverID'] = sID then
      redQ2.Lines.Add(tblBookings['BookingNum'] +#9 +
              DateToStr(tblBookings['BookingDate']));
    tblBookings.Next;
   end;
end;
// =====================================================================
```

```
//                        Question 2.2.3 – 4 marks
// =====================================================================
procedure TfrmQuestion2.btnQ2_2_3Click(Sender: TObject);
begin
//Question 2.2.3
  tblLDrivers.Insert;
  tblLDrivers['LDriverID']:= '0405060708091';
  tblLDrivers['LDriverName'] :='Trish';
  tblLDrivers['LDriverSurname'] := 'Malope';
  tblLDrivers['LDriverCell']  :='0710810911';
  tblLDrivers.Post;

//Provided code
  ShowMessage('Learner driver has been added.');
end;


{$REGION DB CONNECTION}
//Setup DB connections - DO NOT CHANGE!


// =====================================================================
procedure TfrmQuestion2.bmbRestoreDBClick(Sender: TObject);
begin
  // Restores the Database
  dbCONN.RestoreDatabase;
  dbCONN.setupGrids(dbgBookings, dbgSQL);
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
  //Format rich edit
  redQ2.Paragraph.TabCount := 2;
  redQ2.Paragraph.Tab[0] := 70;
  redQ2.Paragraph.Tab[1] := 100;
  // Sets up the connection to database and opens the tables.
  dbCONN := TConnection.Create;
  dbCONN.dbConnect;
  tblLDrivers := dbCONN.tblOne;
  tblBookings := dbCONN.tblMany;
  dbCONN.SetupGrids(dbgBookings, dbgSQL);
  pgcTabs.ActivePageIndex := 0;

end;
// =====================================================================

{$ENDREGION}
end.
```

ANNEXURE G: SOLUTION FOR QUESTION 3

Object class

```
// =======================================================================
//                       Question 3.1.1 – 4 marks
// =======================================================================
function TDeliveryTrip.compileTripNum: Integer;
var
  iTripNum : Integer;
begin
 //Question 3.1.1
  repeat
    iTripNum := randomRange(100,1000);
  until iTripNum MOD 10 <> 0;
  result := iTripNum;
end;
// =======================================================================
//                       Question 3.1.2 – 5 marks
// =======================================================================
constructor TDeliveryTrip.create(sDeparture, sDestination:String;
rLoad:Real);
begin
    fDeparture := sDeparture;
    fDestination:= sDestination;
    fLoad := rLoad;
    fDistance := 0;
    fTripNumber := compileTripNum;
end;
// =======================================================================
//                       Question 3.1.3 – 2 marks
// =======================================================================
function TDeliveryTrip.getDistance: integer;
begin
    result := fDistance;
end;
// =======================================================================
//                       Question 3.1.4 – 2 marks
// =======================================================================
procedure TDeliveryTrip.setDistance(iDist: integer);
begin
   fDistance := iDist;
end;
// =======================================================================
//                       Question 3.1.5 – 8 marks
// =======================================================================
function TDeliveryTrip.determineTruckType: string;
begin
   if fDistance = 0 then
     result := 'No distance'
    else
        if (fLoad <= 1000) then
         result := 'Light truck'
          else if fLoad <= 5000 then
               result := 'Medium truck'
              else result := 'Heavy truck';
end;
```

```
// ====================================================================
//                         Provided Code
// ====================================================================

function TDeliveryTrip.getDestination: String;
begin
  Result := fDestination;
end;

function TDeliveryTrip.getDeparture: String;
begin
  Result := fDeparture;
end;

function TDeliveryTrip.toString: String;
begin
 Result :='Trip number: '+ IntToStr(fTripNumber) + #13 +
          'Departure: ' + fDeparture + #13 +
          'Destination: '+ fDestination + #13 +
          'Load: '+ FloatToStrF(fLoad,ffFixed,5,1) +' kg'+ #13 +
          'Distance: ' + IntToStr(fDistance) + ' km' + #11+ '';
end;

//end of provided code

end.
// ====================================================================
```

Main Form Unit

```
// =====================================================================
//                    Question 3.2.1 – 5 marks
// =====================================================================
procedure TfrmQuestion3.btnQ3_2_1Click(Sender: TObject);
var
    sDepart, sDestination : String;
    rLoad: real;

begin
 //Provided code
  redQ3.Clear;

 //Code Question 3.1 here
  sDepart := rgpQ3_2_1_Departure.Items[rgpQ3_2_1_Departure.ItemIndex];
  sDestination :=
     rgpQ3_2_1_Destination.Items[rgpQ3_2_1_Destination.ItemIndex];
  rLoad := StrToFloat(edtQ3_2_1.Text);

  objTrip:= TDeliveryTrip.create(sDepart, sDestination, rLoad);
  redQ3.Lines.Add(objTrip.toString);
end;
// =====================================================================
//                    Question 3.2.2 – 12 marks
// =====================================================================
procedure TfrmQuestion3.btnQ3_2_2Click(Sender: TObject);
var
  sLine, sTown, sDep, sDest : String;
  iPos : Integer;
  iDistance: integer;
  myFile : TextFile;

begin
 //Provided code
  redQ3.Clear;
  if FileExists('DataQ3.txt') <> True then
    begin
      ShowMessage('File Does not Exist');
      Exit;
    end
  else

  //Code Question 3.2 here
  AssignFile(myFile,'DataQ3.txt');
  Reset(myFile);
  while not EOF(myFile) do
  begin
    readln(myFile,sLine);
    sDep := objTrip.getDeparture;
    sDest := objTrip.getDestination;
    if (Pos(sDest, sLine) > 0) AND (Pos(sDep, sLine) > 0) then
        begin
           iPos := pos('#',sLine);

           objTrip.setDistance(StrToInt(copy(sLine, iPos + 1)));
```

```
            edtQ3_2_2.Text := IntToStr(objTrip.getDistance);
        end;
   end;
   CloseFile(myFile);
   redQ3.Lines.Add(objTrip.toString);
end;


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

procedure TfrmQuestion3.btnQ3_2_3Click(Sender: TObject);
begin
    //Code Question 3.2.3 here
    edtQ3_2_3.Text := objTrip.determineTruckType;
end;

end.
```

ANNEXURE H: SOLUTION FOR QUESTION 4

```
unit Question4_u;

interface

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

type
  TfrmQuestion4 = class(TForm)
    pnlQ4: TPanel;
    Panel2: TPanel;
    redQ4: TRichEdit;
    edtQ4: TEdit;
    cmbQ4: TComboBox;
    lblQ4_2_2: TLabel;
    grbQ4_2: TGroupBox;
    lblQ4_2_1: TLabel;
    btnQ4_1: TButton;
    GroupBox2: TGroupBox;
    btnQ4_2: TButton;
    procedure btnQ4_1Click(Sender: TObject);
    procedure cmbQ4Enter(Sender: TObject);
    procedure btnQ4_2Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  frmQuestion4: TfrmQuestion4;


  arrRegNumbers: array [1 .. 20] of String = (
    'CA 123 456','NN 21514',
    'BBC 123 MP','BEC 558 EC',
    'XRG 123 L', 'CA JN 912 WP',
    'CD 083 027','CX 55472',
    'BCD 123 MP','ND 122 156',
    '786 ZN',    'SNH 582 GP',
    'IXLR8 NM',  'JJO 114 MP',
    'OQE 329 GP','ALP 439 GP',
    'CAA 220 002','YTF 871 EC',
    'WIL 007 GP','CFA 1001'
  );

  arrEntryTimes: array [1 .. 20] of String = (
    '08:00',  '09:22',
    '10:11',  '10:15',
    '10:43',  '11:03',
    '11:34',  '12:19',
    '12:32',  '12:45',
```

```
    '12:59',  '13:03',
   '13:20',  '14:24',
   '14:36',  '15:41',
   '15:51',  '16:06',
   '16:38',  '17:48'
  );

iCounter:Integer;
implementation

{$R *.dfm}
// ===================================================================
//                      Question 4.1 – 7 Marks
// ===================================================================

procedure TfrmQuestion4.btnQ4_1Click(Sender: TObject);
var
  sLine: String;
  I: integer;
  J: integer;
begin
//Provided code
 redQ4.Clear;
 redQ4.Lines.Add('#'+ #9 + 'RegNumber' + #9 + 'Time In');

// Code Question 4.1 here
 for I:= 1 to iCounter do
   redQ4.Lines.Add(IntToStr(I) + #9 + arrRegNumbers[I] + #9+
arrEntryTimes[I]);

 {Alternative
 redQ4.Lines.Add(format('%-5s%-15s%10s', ['#', 'Number Plate', 'Time
In']));
 for I := 1 to iCounter do
   redQ4.Lines.Add(format('%-5d%-15s%10s',
   [I, arrNumberPlates[I], arrEntryTimes[I]])); }
end;


// ===================================================================
//                      Question 4.2 – 23 Marks
// ===================================================================

procedure TfrmQuestion4.btnQ4_3Click(Sender: TObject);
var
  sLine, sRegNum, sTimeIn, sTimeOut: String;
  iIndex, iTimeInInMins, iTimeOutInMins, iTimeSpent: integer;
  iHoursMin, iMinutes, iPosColon : Integer;
  I, iCounter: integer;
  rTariff, rCost:real;
begin
 redQ4.Clear;
 for I := 1 to iCounter do
  if cmbQ4.Text = arrRegNumbers[I] then
    begin
      iIndex := I;
      sRegNum := arrRegNumbers[I];
      sTimeIn := arrEntryTime[I];
    end;
```

```delphi
    sTimeOut := edtQ4.Text;
    iPosColon := pos(':',sTimeOut);
    iHoursMin := StrToInt(copy(sTimeOut, 1, 2)) * 60;
    iMinutes := StrToInt(copy(sTimeOut, iPosColon + 1, 2));
    iTimeOutInMins := iHoursMin + iMinutes;

    iPosColon := pos(':',sTimeIn);
    iHoursMin := StrToInt(copy(sTimeIn, 1, 2)) * 60;
    iMinutes := StrToInt(copy(sTimeIn, iPosColon + 1, 2));
    iTimeInInMins := iHoursMin + iMinutes;

    iTimeSpent := iTimeOutInMins - iTimeInInMins;

    if iTimeSpent > 0 then
     begin

     case iTimeSpent of
      0 .. 30:    rTariff := 0;
      31 .. 120:  rTariff := 50;
      121 .. 240: rTariff := 40;
     else
      rTariff := 30;
    end;

 rCost := rTariff * Ceil(iTimeSpent / 60);

 for I := iIndex to length(arrRegNumbers) - 1 do
  begin
    arrRegNumbers[I] := arrRegNumbers[I + 1];
    arrEntryTime[I] := arrEntryTime[I + 1];
  end;
Dec(iCounter);

 redQ4.Lines.Add('Registration number: ' + sRegNum);
 redQ4.Lines.Add('Entry time: ' + sTimeIn);
 redQ4.Lines.Add('Exit time: ' + sTimeOut);
 redQ4.Lines.Add('Time spent: ' + IntToStr(iTimeSpent div 60) + ' hours ' +
                          IntToStr(iTimeSpent mod 60) + ' min');
 redQ4.Lines.Add('Tariff per hour: ' +
                 floatToStrF(rTariff, ffCurrency, 10, 2));
 redQ4.Lines.Add('Cost of parking: ' +
                 floatToStrF(rCost, ffCurrency, 10, 2));
end
else
    redQ4.Lines.Add('Invalid exit time');
end;

//PROVIDED CODE - DO NOT CHANGE
procedure TfrmQuestion4.cmbQ4Enter(Sender: TObject);
var
  I: integer;
begin
  cmbQ4.Clear;
  for I := 1 to length(arrRegNumbers) do
  begin
    cmbQ4.Items.Add(arrRegNumbers[I]);
  end;

end;
```

               Please turn over

```
procedure TfrmQuestion4.FormActivate(Sender: TObject);
begin
  redQ4.Paragraph.TabCount:= 2;
  redQ4.Paragraph.Tab[0] := 50;
  redQ4.Paragraph.Tab[1] := 150;
end;

end.
```