basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE12

INFORMATION TECHNOLOGY P1

NOVEMBER 2016

MEMORANDUM

MARKS: 150

This memorandum consists of 32 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 were not followed or the requirements of the question were not met.

- **Annexures A, B and C** (pages 3-10) include the marking grid for each question for using either one of the two programming languages.

- **Annexures D, E and F** (pages 11-19) contain examples of solutions for Java for QUESTION 1 to QUESTION 3 in programming code.

- **Annexures G, H and I** (pages 20-32) contain examples of solutions for Delphi for QUESTION 1 to QUESTION 3 in programming code.

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

ANNEXURE A:

SECTION A:

QUESTION 1: MARKING GRID- GENERAL PROGRAMMING SKILLS

CENTRE NUMBER:	**EXAMINATION NUMBER:**		
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
	A learner must be penalised only once if the same error is repeated.		
1.1	Extract length from the text box ✓, convert to real value ✓ Extract width from the text box and convert to value ✓ Extract height from the text box and convert to value *Calculate volume using the formula* Volume = length * width * height ✓ Display the volume in the text box ✓	**5**	
1.2	**Button - [Question 1_2]** Volume = volume * 1000 if volume <=500 ✓ *Calculate cost using the formula* Cost = Volume * 0.25 ✓ else if volume <= 800 ✓ *Calculate cost using the formula* Cost = 500 * 0.25 ✓+ (volume - 500) * 0.35 ✓ else ✓ *Calculate cost using the formula* Cost = 500 * 0.25 +300* 0.35 ✓+ (volume - 800) * 0.45 ✓ Display the cost in the text box ✓ Concepts: • 2 marks :if volume <= 500 and calculation • 3 marks: if volume more than 500 and less than or equal to 800 and calculation • 1 mark : else/if • 2 marks: if volume > 800 and calculation • 1 mark: display	**9**	

1.3	**Button - [Question 1_3]** Obtain the lifespan in months from the text box as a value ✓ *Calculate the number of months by mod or any other method* months = lifespan modulus 12 ✓ C*alculate the number of years using the correct formula* years = lifespan divide by 12 ✓ (discard the fraction value✓) Display the years ✓and months ✓	**6**	
1.4	**Button - [Question 1_4]** Extract the setup cost from the text box and convert to value ✓ Extract the income for first year from the text box and convert to value ✓ Display headings in output area ✓ Initialise yearNumber variable to 1 ✓ Use a loop to check the setupCost > 0 ✓ setupCost = setupCost – yearly income ✓ if setupCost > 0 ✓ display the yearNumber, yearly income, setupCost ✓ else display the yearNumber, yearly income, “Paid off” ✓ Increase yearNumber by 1 ✓ Increase by 10% ✓the yearly income ✓ All monetary values must be formatted to currency✓ with two decimal places ✓ *Alternate solution:* Extract the setup cost from the text box and convert to value (1mark) Extract the income for first year from the text box and convert to value (1mark) Display headings in output area (1mark) Initialise yearNumber variable to 1 (1mark) Use a loop to check the setupCost > Income (2 marks) setupCost = setupCost – yearly income (1mark) display the yearNumber, yearly income, setupCost (1mark) Increase yearNumber by 1 (1mark) Increase by 10% (1mark) the yearly income (1mark) display the yearNumber, yearly income, “Paid off” (1mark) All monetary values must be formatted to currency(1mark) with two decimal places (1mark)	**14**	

1.5	**Button - [Question 1_5_1]** Generate a random number ✓ for dice 1 in correct range ✓ Generate a random number for dice 2 ✓ Display dice 1 label and value in output area } ✓ Display dice 2 label and value in output area } *Check if the numbers on dice 1 and dice 2 are consecutive* Test: if dice 1 – dice 2 = 1 ✓ Test: OR if dice 2 – dice 1 = 1 ✓ *Alternate solution:* If ABS(Dice1 – Dice2) = 1 (2 marks) Enable the radio buttons or radio group components✓ Enable button Question 1_5_2 ✓	**8**	
	Button - [Question 1_5_2] Use a dialog box to enter the ticket number ✓ Extract the system date ✓✓ Select option from radio button ✓ Compile the reference number by joining: the ticket number and the date ✓ the first two letters of the selected course in uppercase ✓ # in correct positions ✓ Display the reference number in the output area ✓	**8**	
	TOTAL:	**50**	

ANNEXURE B:

SECTION B:

QUESTION 2: MARKING GRID - OBJECT-ORIENTED PROGRAMMING

CENTRE NUMBER:	**EXAMINATION NUMBER:**		
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
2.1.1	**Mutator method for setVisitDate:** Method definition with date parameter ✓ Assign the parameter value to the attribute ✓	**2**	
2.1.2	**requireTourGuide method:** Correct return **string** data type ✓ Check if tourGuide = true ✓ Return ✓ "Yes" ✓(*Correct word*) else Return "No"	**4**	
2.1.3	**isConfirmed method:** Method definition with parameter and correct return **Boolean** data type ✓ Check if dayTotal + groupSize ✓ <= 500 ✓ Return true ✓ else Return false ✓	**5**	
2.1.4	**calcAmount method** Method definition with 2 correct parameters ✓ *Calculate how many learners have free entrance:* numberFree = groupSize divided by 10 to give whole number ✓ *Calculate cost:* amount = (groupSize – numberFree) ✓ * costPerPerson ✓ Check if tour guide required ✓ amount = amount + costTourGuide ✓ Return amount ✓	**7**	
2.1.5	**toString method:** Label each item on new line ✓ All the correct attributes/get-methods ✓ Call the method to confirm tour guide ✓ Return the compiled string ✓	**4**	

2.2.1	**Button - [2.2.1 Instantiate object]** Extract the school name from the text box ✓ Extract the date from the list box ✓ Extract the size of group from the text box and convert to integer type ✓ If the check box is ticked ✓ Set the tourGuide to true else Set the tourGuide to false ✓ Instantiate object with correct parameters ✓ Display message to indicate that object is instantiated ✓	**7**	
2.2.2(a)	**Button – [2.2.2 – Confirm availability]** **determineDayTotal** method with date parameter ✓ Set dayTotal to 0 ✓ *Read and separate information in text file* {*Delphi: AssignFile, Reset* *Java: Create object to read from file*} ✓✓ Loop through the file ✓ Read a line at a time ✓ Find the position of the date ✓ Identify the date of visit from the line ✓ Extract the group size from the line ✓ Check if date from line = date received as parameter ✓ Increase dayTotal by groupSize ✓ CloseFile(TxtFile); Return the final dayTotal ✓	**12**	

2.2.2(b)	Using the method **determineDayTotal** to obtain the value for the dayTotal ✓ *Send the dayTotal as argument to the isConfirmed method of the object class* Check if the isConfirmed method returns true ✓ Call the calcAmount method ✓ with 2 constant values as arguments ✓ to calculate the amount Display string returned by the toString method ✓ and amount value returned by calcAmount method ✓ Else ✓ Display a message to indicate no availability ✓ Show the panel called pnlAvailability ✓ Loop through the list box ✓ Extract date from the list box ✓ Call the determineDayTotal method using the date extracted as an argument ✓ Pass the dayTotal ✓ to the isConfirmed method ✓ If the isConfirmed method returns true ✓ Add the date to the items of the combo box ✓	**16**	
2.2.3	**Button – [2.2.3 – Confirm new date]** Check if there are any items in the combo box ✓ Extract selected date from combo box ✓ Call the setVisitDate method with this date as argument ✓ Call the toString and the calcAmount method to display ✓ Else Display a suitable message indicating the application to go on excursion was unsuccessful ✓	**5**	
	TOTAL:	**62**	

ANNEXURE C:

SECTION C:

QUESTION 3: MARKING GRID – PROBLEM SOLVING

CENTRE NUMBER:	**EXAMINATION NUMBER:**		
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
3.1	**Button [3.1 – Activity/Facility codes for all terminals and directions]** Display column headings ✓ Outer loop to control the rows ✓ Join terminal number to line ✓ Inner loop to control the columns ✓ Find correct data element in twoD array ✓ Join data from twoD array to line ✓ End inner loop Display the line in the output area on a new line ✓ End outer loop Display in neat columns ✓	**9**	
	Use a data structure or any other way to find the directions North, South, East and West ✓		
3.2	**Button** **[3.2 – Activities/Facilities from a selected terminal and direction]** Display the terminal number ✓ and the direction in the output area ✓ Extract the code from the twoD array at this terminal number ✓ and in this direction ✓ Loop through the length of the extracted code ✓ Loop through the arrCodes array ✓ Compare each letter in the code ✓ to the values in the arrCodes array ✓ Display the activity from the arrActivities array✓ at the same index ✓ as in the arrCodes ✓ array	**11**	

3.3	**Button [3.3 – Access routes to selected activity/ facility]** Extract index of the activity/facility selected in the combo box ✓ Set counter to 0 ✓ Display heading ✓ Find the code of selected activity ✓ Outer loop for rows ✓ Inner loop for columns ✓ Test if the code of the selected activity is a part of the code in the twoD array ✓ Display the terminal number and the direction ✓ Increase counter by 1 ✓ Display the label for the number of activities and the value of counter ✓	**10**	
3.4	**Button** **[3.4 – Maintenance at a selected activity/facility]** Extract index of the activity/facility selected in the combo box ✓ Outer loop for rows ✓ Inner loop for columns ✓ Check if the code of the activity/facility selected in the combo box is a part of the code in the twoD array ✓ Delete the code letter from the twoD array ✓ at correct row and correct column ✓ Display a message indicating the access routes to the selected activity/facility is closed ✓ Call a method to display the updated twoD array with headings ✓	**8**	
	TOTAL	**38**	

SUMMARY OF LEARNER'S MARKS:

CENTRE NUMBER:		**EXAMINATION NUMBER:**		
	SECTION A	**SECTION B**	**SECTION C**	
	QUESTION 1	**QUESTION 2**	**QUESTION 3**	**GRAND TOTAL**
MAX. MARKS	**50**	**62**	**38**	**150**
LEARNER'S MARKS				

ANNEXURE D: SOLUTION FOR QUESTION 1: JAVA

```
// Provided code
SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");

// Global variables
      DecimalFormat df1 = new DecimalFormat("0.00"); // Q1.1
      DecimalFormat df = new DecimalFormat("R0.00"); // Q1.2
      double volume = 0; // Q1.1 & Q1.2
//=====================================================================
//Question 1.1
//=====================================================================
private void btnQues11ActionPerformed(java.awt.event.ActionEvent evt) {
        double length,breadth,height;
        length = Double.parseDouble(txfLength.getText());
        width = Double.parseDouble(txfWidth.getText());
        height = Double.parseDouble(txfHeight.getText());
        volume = length * width * height;
        txfVolume.setText("" + df1.format(volume));
    }
//=====================================================================
//Question 1.2
//=====================================================================
 private void btnQues12ActionPerformed(java.awt.event.ActionEvent evt) {
       double cost = 0;
      volume = volume * 1000;
        if (volume <=500) {
          cost = volume * 0.25 ;
        }
         else if (volume <=800) {
          cost = (500 * 0.25)+ (volume- 500) * 0.35 ;
        }
         else{
             cost = (500 * 0.25)+ (300 * 0.35)+(volume- 800) * 0.45 ;
         }
        txfCost.setText(""+df.format(cost));
    }
//=====================================================================
//Question 1.3
//=====================================================================
 private void btnQues13ActionPerformed(java.awt.event.ActionEvent evt) {
       int lifespanInMonths =
                  Integer.parseInt(txfLifespanInMonths.getText());
       int months = lifespanInMonths % 12;
       int years = (lifespanInMonths - months) / 12;
       txfYearsAndMonths.setText(years+" years and "+months+" months");
    }
//=====================================================================
//Question 1.4
//=====================================================================
 private void btnQues14ActionPerformed(java.awt.event.ActionEvent evt) {
   txaQ14.setText(String.format("%-10s%-14s%-
                  15s\n","Year","Income","Balance"));
         double setupCost = Double.parseDouble(txfSetupCost.getText());
         double yearlyIncome =
               Double.parseDouble(txfIncomeYear1.getText());
         int yearNumber  = 1;
         while (setupCost > 0) {
             setupCost= setupCost - yearlyIncome;
             if (setupCost > 0) {
                 txaQ14.append(String.format("%-10sR%-13.2fR%-
                  13.2f\n",yearNumber,yearlyIncome,setupCost));
```

```
            }
            else{
             txaQ14.append(String.format("%-10sR%-13.2f%-
             15s\n",yearNumber,yearlyIncome,"Paid off"));
            }
            yearNumber++;
            yearlyIncome = yearlyIncome * 1.1;
        }
    }

//=====================================================================
//Question 1.5.1
//=====================================================================

private void btnQues151ActionPerformed(java.awt.event.ActionEvent evt) {
        int dice1 = (int) (Math.random() * 6) + 1;
        int dice2 = (int) (Math.random() * 6) + 1;
        txaQues15.setText("Dice 1 = " + dice1);
        txaQues15.append("\nDice 2 = " + dice2);
        if ((dice1 == (dice2 + 1))  || (dice1 == (dice2 - 1))) {
            rgbSnorkelling.setEnabled(true);
            rgbSwimming.setEnabled(true);
            btnQues152.setEnabled(true);
        }
        else{
            rgbSnorkelling.setEnabled(false);
            rgbSwimming.setEnabled(false);
            btnQues152.setEnabled(false);
        }
    }
//=====================================================================
//Question 1.5.2
//=====================================================================
 private void btnQues152ActionPerformed(java.awt.event.ActionEvent evt) {
       String ticketNum = JOptionPane.showInputDialog("Enter your ticket
       number");
       Date now = new Date();
       String date = sdf.format(now);
       String referenceNum = ticketNum +"#"+date+"#";
        if (rgbSnorkelling.isSelected()) {
            referenceNum = referenceNum +
                  rgbSnorkelling.getText().substring(0,2);
        }
        else{
             referenceNum = referenceNum
                  + rgbSwimming.getText().substring(0,2);
        }
       txaQues15.append("\nReference number:\n"+referenceNum.toUpperCase());
 }
```

ANNEXURE E: SOLUTION FOR QUESTION 2: JAVA

SOLUTION FOR QUESTION 2: OBJECT CLASS

```
public class Excursion {

//=====================================================================
//Given code
//=====================================================================
   private String schoolname, visitDate;
   private int groupSize;
   private boolean tourGuide;

    public Excursion(String schoolName, String visitDate,
                     int groupSize, boolean tourGuide) {
        this.schoolName = schoolName;
        this.visitDate = visitDate;
        this.groupSize = groupSize;
        this.tourGuide = tourGuide;
    }

    public String getSchoolName() {
        return schoolName;
    }

    public String getVisitDate() {
        return visitDate;
    }

    public int getGroupSize() {
        return groupSize;
    }
//=====================================================================
//Question 2.1.1
//=====================================================================
    public void setVisitDate(String visitDate) {
        this.visitDate = visitDate;
    }
//=====================================================================
//Question 2.1.2
//=====================================================================
   public String requireTourGuide(){
       if (tourGuide){
           return "Yes";
       }
       else{
           return "No";
       }
   }

//=====================================================================
//Question 2.1.3
//=====================================================================
   public boolean isConfirmed(int dayTotal){
       if ((dayTotal + groupSize) <=500 ) {
           return true;
       }
       else{
           return false;
       }   }
```

```
//============================================================
//Question 2.1.4
//============================================================
   public double calcAmount(double personCost,double guideCost){
     int  numberFree = groupSize / 10;
     double amount = (groupSize - numberFree)  * personCost;
       if (tourGuide) {
           amount += guideCost;
       }
       return amount;
   }

//============================================================
//Question 2.1.5
//============================================================
   public String toString(){
       return "School name: " + schoolName + "\nDate of visit: "+ visitDate+
            "\nNumber of learners: "+ groupSize + "\nTour guide: "+
requireTourGuide();
   }
}
```

GUI CLASS: QUESTION2_SOLUTION

```
//============================================================================
//Provided code
//============================================================================
public class Question2 extends javax.swing.JFrame {

public Question2() {
        initComponents();
        this.setLocationRelativeTo(this);
        pnlAvailability.setVisible(false);
}

Excursion objExcursion;
final double costPerPerson = 75.00;
final double costTourGuide = 300.00;

//============================================================================
//Question 2.2.1
//============================================================================
private void btnQues221ActionPerformed(java.awt.event.ActionEvent evt) {
   String school = txfSchoolname.getText();
   String date = ""+lstVisitDate.getSelectedValue();
   int groupSize = Integer.parseInt(txfGroupSize.getText().trim());
   boolean tourGuide = false;
   if (chkTourGuide.isSelected()) {
       tourGuide = true;
   }

   objExcursion= new Excursion(school, date, groupSize, tourGuide);
   JOptionPane.showMessageDialog(null,"Excursion object has been
          Instantiated.");
   pnlAvailability.setVisible(false);
  }
//============================================================================
//Question 2.2.2 (a)
//============================================================================
public int determineDayTotal(String dateOfVisit){
   int total = 0;
   try {
     Scanner sc = new Scanner (new FileReader("DataQ2.txt"));
     while (sc.hasNext()) {
       String line = sc.nextLine();
       String [] arrPart = line.split("#");
       String schoolName = arrPart[0];
       String date = arrPart[1];
       int groupSize = Integer.parseInt(arrPart[2]);
       if (date.equalsIgnoreCase(dateOfVisit)) {
              total += groupSize;
       }//if
     }//while
    sc.close();
   } catch (Exception e) {
       JOptionPane.showMessageDialog(null,"Error:"+e.getMessage());
  }
  return total;
 }
```

```
//=========================================================================
//Question 2.2.2 (b)
//=========================================================================
private void btnQues222ActionPerformed(java.awt.event.ActionEvent evt) {
  int total = determineDayTotal(objExcursion.getVisitDate());
  boolean successful = false;
  if (objExcursion.isConfirmed(total)) {
     JOptionPane.showMessageDialog(null,objExcursion.toString()+"\nAmount to be
       paid: "+df.format(objExcursion.calcAmount(costPerPerson,
                 costTourGuide)));
        successful = true;
      }
   else{
      JOptionPane.showMessageDialog(null,"There is no space on the
                  date selected." );
      pnlAvailability.setVisible(true);
      cmbAvailableDates.removeAllItems();
      for (int i = 0; i < 5; i++) {
           lstVisitDate.setSelectedIndex(i);
           String date = ""+lstVisitDate.getSelectedValue();
           System.out.println(date);
           total = determineDayTotal(date);
           if (objExcursion.isConfirmed(total)) {
               cmbAvailableDates.addItem(""+date);
           }//if
       }//for
    }//else
 }
//=========================================================================
//Question 2.2.3
//=========================================================================
private void btnQues223ActionPerformed(java.awt.event.ActionEvent evt) {
  if (cmbAvailableDates.getItemCount() > 0) {
       objExcursion.setVisitDate(""+cmbAvailableDates.getSelectedItem());
       JOptionPane.showMessageDialog(null,objExcursion.toString()+"\nAmount to
            be paid: "+df.format(objExcursion.calcAmount(costPerPerson,
              costTourGuide)));
  }
  else{
     JOptionPane.showMessageDialog(null, "The application for " +
          objExcursion.getSchoolName()  + " is unsuccessful.");
  }
 }
```

ANNEXURE F: SOLUTION FOR QUESTION 3: JAVA

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

int terminal = 0;
int direction = 0;
    char arrCodes[] = {'W',
        'A',
        'S',
        'R',
        'X',
        'D',
        'H',
        'P',
        'T',
        'L'};
    String[] arrActivities = {
        "Water park",
        "Aquarium",
        "Sea",
        "Restaurants",
        "Shopping",
        "Diving",
        "Help desk",
        "Penguin park",
        "Shark tank",
        "Dolphin shows"
    };
    String[][] arrActCodes = {{"DXWAT", "HRDST","STWLP", "RDT"},
                              {"SWA", "SRXD", "LWXH", "SHA"},
                              {"WLSR", "AT", "DATX", "HW"}};


private void btnTerminal1ActionPerformed(java.awt.event.ActionEvent evt) {
        terminal = 0;
    }

    private void btnTerminal2ActionPerformed(java.awt.event.ActionEvent evt) {
        terminal = 1;
    }

    private void btnTerminal3ActionPerformed(java.awt.event.ActionEvent evt) {
        terminal = 2;
    }

    private void btnNorthActionPerformed(java.awt.event.ActionEvent evt) {
        direction = 0;
    }

    private void btnSouthActionPerformed(java.awt.event.ActionEvent evt) {
        direction = 1;
    }

    private void btnEastActionPerformed(java.awt.event.ActionEvent evt) {
        direction = 2;
    }

    private void btnWestActionPerformed(java.awt.event.ActionEvent evt) {
        direction = 3;
    }
```

```
//=========================================================================
// Global variables
//=========================================================================
    String[] arrDirections = {
        "North",
        "South",
        "East",
        "West"
    };

//=========================================================================
//Question 3.1
//=========================================================================
private void btnQues31ActionPerformed(java.awt.event.ActionEvent evt) {
        txaQues3.append(String.format("%-15s%-10s%-10s%-10s%-10s\n", "",
                                      "North", "South", "East", "West"));
        for (int row = 0; row < 3; row++) {
            txaQues3.append(String.format("%-15s","Terminal "+(row+1)));
            for (int col = 0; col < 4; col++) {
                   txaQues3.append(String.format("%-
                   10s",arrActCodes[row][col]));
            }
            txaQues3.append("\n");

        }
    }
//=========================================================================
//Question 3.2
//=========================================================================
private void btnQues32ActionPerformed(java.awt.event.ActionEvent evt) {
        txaQues3.setText("Terminal "+ (terminal+1)+", "+
        arrDirections[direction]+"\n");
        String codes = arrActCodes[terminal][direction];
        for (int i = 0; i < codes.length(); i++) {
            for (int j = 0; j < arrCodes.length; j++) {
                if (codes.charAt(i)==arrCodes[j]) {
                    txaQues3.append(arrActivities[j]+"\n");
                }
            }
        }
    }
//=========================================================================
//Question 3.3
//=========================================================================
private void btnQues33ActionPerformed(java.awt.event.ActionEvent evt) {
        int index = cmbQues3.getSelectedIndex();
        int count  = 0;
        txaQues3.setText("Access routes to
         "+cmbQues3.getSelectedItem()+"\n");

        for (int iRow = 0; iRow < 3; iRow++) {
            for (int iCol = 0; iCol < 4; iCol++) {
                if (arrActCodes[iRow][iCol].indexOf(arrCodes[index])
                  >= 0) {
                    txaQues3.append("Terminal "+(iRow+1)+",
                    "+arrDirections[iCol]+"\n");
                    count++;
                }
            }
        }
        txaQues3.append("\nNumber of access routes: "+count);
    }
```

```
//============================================================================
//Question 3.4
//============================================================================
private void btnQues34ActionPerformed(java.awt.event.ActionEvent evt) {
         txaQues3.setText("Updated information:\n\n");
         int index = cmbQues3.getSelectedIndex();
           for (int iRow = 0; iRow < 3; iRow++) {
               for (int iCol = 0; iCol < 4; iCol++) {
                   if (arrActCodes[iRow][iCol].indexOf(arrCodes[index])
                   >=0) {
                     arrActCodes[iRow][iCol]=
            arrActCodes[iRow][iCol].replaceAll(arrCodes[index]+"","");
                   }
               }
           }
               JOptionPane.showMessageDialog(null,"The access routes to
               "+arrActivities[index]+" are closed.");
               btnQues31.doClick();
    }
```

ANNEXURE G: SOLUTION FOR QUESTION 1: DELPHI

```
=========================================================================
// Provided coded
=========================================================================

unit Question1_U;

interface

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

type
  TfrmQuestion1 = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    GroupBox1: TGroupBox;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    edtLength: TEdit;
    edtWidth: TEdit;
    edtHeight: TEdit;
    btnQues11: TButton;
    edtVolume: TEdit;
    GroupBox2: TGroupBox;
    btnQues12: TButton;
    edtCost: TEdit;
    GroupBox3: TGroupBox;
    btnQues13: TButton;
    Label4: TLabel;
    edtQues13: TEdit;
    GroupBox4: TGroupBox;
    btnQues14: TButton;
    redQues14: TRichEdit;
    GroupBox5: TGroupBox;
    btnQues151: TButton;
    redQues15: TRichEdit;
    rgpPrizes: TRadioGroup;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    edtLifespan: TEdit;
    lblVolume: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    edtInitialCost: TEdit;
    edtIncome: TEdit;
    btnQues152: TButton;
    procedure btnQues11Click(Sender: TObject);
    procedure btnQues12Click(Sender: TObject);
    procedure btnQues13Click(Sender: TObject);
    procedure btnQues14Click(Sender: TObject);
    procedure btnQues151Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure btnQues152Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
var
  frmQuestion1: TfrmQuestion1;

//=====================================================================
//Global variable (Q1.1 & Q1.2)
//=====================================================================

  rVolume: real;

implementation

{$R *.dfm}
//=====================================================================
//Question 1.1
//=====================================================================

procedure TfrmQuestion1.btnQues11Click(Sender: TObject);
var
  rLength, rBreadth, rHeight: real;
begin
  rLength := StrToFloat(edtLength.Text);
  rWidth := StrToFloat(edtWidth.Text);
  rHeight := StrToFloat(edtHeight.Text);
  rVolume := rLength * rWidth * rHeight;
  edtVolume.Text := FloatToStrF(rVolume, ffFixed, 6,2);
end;
//=====================================================================
//Question 1.2
//=====================================================================
procedure TfrmQuestion1.btnQues12Click(Sender: TObject);
var
  rCost: real;
begin
  rVolume:= rVolume * 1000;
  if rVolume <= 500 then
    rCost := rVolume * 0.25
  else if rVolume <= 800 then
    rCost := 500 * 0.25 + (rVolume - 500) * 0.35
  else
    rCost := 500 * 0.25 + 300 * 0.35 + (rVolume - 800) * 0.45;

  edtCost.Text := FloatToStrF(rCost, ffCurrency, 6, 2);
end;
//=====================================================================
//Question 1.3
//=====================================================================
procedure TfrmQuestion1.btnQues13Click(Sender: TObject);
var
  iLifespanInMonths, iYears, iMonths: integer;
begin

  iLifespanInMonths := StrToInt(edtLifespan.Text);
  iMonths := iLifespanInMonths mod 12;
  iYears := (iLifespanInMonths - iMonths) div 12;
  edtQues13.Text := IntToStr(iYears) + ' years and ' + IntToStr(iMonths)
    + ' months';
end;
```

```
//=====================================================================
//Question 1.4
//=====================================================================
procedure TfrmQuestion1.btnQues14Click(Sender: TObject);
var
  rSetUpCost, rYearlyIncome: real;
  iYearNumber: integer;
begin
  redQues14.Paragraph.TabCount := 2;
  redQues14.Paragraph.Tab[0] := 40;
  redQues14.Paragraph.Tab[1] := 100;

  redQues14.Lines.Add('Year' + #9 + 'Income' + #9 + 'Balance');

  rSetUpCost := StrToFloat(edtInitialCost.Text);
  iYearNumber := 1;
  rYearlyIncome := StrToFloat(edtIncome.Text);

  while rSetUpCost > 0 do
  begin
    rSetUpCost := rSetUpCost - rYearlyIncome;
    if rSetUpCost > 0 then
      redQues14.Lines.Add(IntToStr(iYearNumber) + #9 + FloatToStrF
          (rYearlyIncome, ffCurrency, 8, 2) + #9 + FloatToStrF
          (rSetUpCost, ffCurrency, 8, 2))
    else
      redQues14.Lines.Add(IntToStr(iYearNumber) + #9 + FloatToStrF
          (rYearlyIncome, ffCurrency, 8, 2) + #9 + 'Paid off');
    Inc(iYearNumber);
    rYearlyIncome := rYearlyIncome * 1.1;
  end;

end;
//=====================================================================
//Question 1.5.1
//=====================================================================
procedure TfrmQuestion1.btnQues151Click(Sender: TObject);
var
  iDice1, iDice2: integer;
begin
  redQues15.Lines.Clear;

  iDice1 := random(6) + 1;
  iDice2 := random(6) + 1;
  redQues15.Lines.Add('Dice 1 = ' + IntToStr(iDice1));
  redQues15.Lines.Add('Dice 2 = ' + IntToStr(iDice2));

  if ((iDice1 = iDice2 + 1) OR (iDice1 = iDice2 - 1)) then
  begin
    btnQues152.Enabled := true;
    rgpPrizes.Enabled := true;
  end
  else
  begin
    btnQues152.Enabled := false;
    rgpPrizes.Enabled := false;
  end;

end;
```

```
//=====================================================================
//Question 1.5.2
//=====================================================================
procedure TfrmQuestion1.btnQues152Click(Sender: TObject);
var
  sTicketNum, sReferenceNum: String;
  dDate: TDateTime;
begin
  sTicketNum := inputbox('Ticket number input', 'Enter your ticket number',
'');
  dDate := Date;
  sReferenceNum := sTicketNum + '#' + DateToStr(dDate);
  sReferenceNum := sReferenceNum + '#' + Uppercase
    (copy(rgpPrizes.Items[rgpPrizes.ItemIndex], 1, 2));
  redQues15.Lines.Add('Reference number: ' + #13 + sReferenceNum);
end;
=======================================================================
//Provided coded
=======================================================================

procedure TfrmQuestion1.FormActivate(Sender: TObject);
begin
  CurrencyString := 'R';
  btnQues152.Enabled := false;
  rgpPrizes.Enabled := false;

end;

end.
```

ANNEXURE H: SOLUTION FOR QUESTION 2: DELPHI

OBJECT CLASS:

```
unit Excursion_U;

interface

uses SysUtils, Math, Messages, Dialogs, DateUtils;

Type
  TExcursion = class(TObject)
  private
    { private declarations }
    fSchoolName, fVisitDate: string;
    fGroupSize: integer;
    fTourGuide: boolean;

  public
    { public declarations }
    constructor Create(sSchoolName:string; sDate:string; iGroupSize:integer;
bTourGuide:boolean);
    procedure setVisitDate(sVisitDate:string);
    function requireTourGuide: string;
    function isConfirmed(iDayTotal:integer): boolean;
    function calcAmount(rPersonCost,rGuideCost:real):real;
    function toString: string;
    function getSchoolName:string;
    function getGroupSize:integer;
    function getVisitDate: string;

  end;

implementation

{ TExcursion }

//=====================================================================
// Provided code for constructor
//=====================================================================

constructor TExcursion.Create(sSchoolName, sDate: string; iGroupSize: integer;
  bTourGuide: boolean);
begin
  fSchoolName := sSchoolName;
  fVisitDate:= sDate;
  fGroupSize:= iGroupSize;
  fTourGuide := bTourGuide;
end;
//=====================================================================
//Question 2.1.1
//=====================================================================

procedure TExcursion.setVisitDate(sVisitDate: string);
begin
  fVisitDate := sVisitDate;
end;
```

```
//====================================================================
//Question 2.1.2
//====================================================================
function TExcursion.requireTourGuide: string;
begin
   if fTourGuide then
     Result := 'Yes'
   else
     Result := 'No';
end;
//====================================================================
//Question 2.1.3
//====================================================================
function TExcursion.isConfirmed(iDayTotal: integer): boolean;
begin
     if ((iDayTotal + fGroupSize)<=500) then
       Result := true
     else
       Result := false;
end;
//====================================================================
//Question 2.1.4
//====================================================================
function TExcursion.calcAmount(rPersonCost,rGuideCost:real): real;
var
  iNumberFree : integer;
  rAmount : real;
begin
   iNumberFree := fGroupSize  div 10;
   rAmount := (fGroupSize - iNumberFree) * rPersonCost;
   if fTourGuide then
       rAmount := rAmount + rGuideCost;
   Result := rAmount;
end;
//====================================================================
//Question 2.1.5
//====================================================================
function TExcursion.toString: string;
begin
  Result := 'School name: ' + fSchoolName + #13 + 'Date of visit: '+
fVisitDate+#13+
            'Number of learners: '+ IntToStr(fGroupSize) + #13 +
            'Tour guide: '+ requireTourGuide;
end;
//====================================================================
// Provided code
//====================================================================
function TExcursion.getSchoolName: string;
begin
  Result := fSchoolName;
end;

function TExcursion.getGroupSize: integer;
begin
  Result := fGroupSize;
end;

function TExcursion.getVisitDate: string;
begin
      Result := fVisitDate;
end;

end.
```

MAIN FORM UNIT: QUESTION2_U.PAS

```
unit Question2_U;

interface

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

type
  TfrmQuestion2 = class(TForm)
    Panel1: TPanel;
    GroupBox1: TGroupBox;
    edtSchoolName: TEdit;
    edtGroupSize: TEdit;
    chbTourGuide: TCheckBox;
    btnQues221: TButton;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    lstVisitDate: TListBox;
    GroupBox3: TGroupBox;
    btnQues222: TButton;
    pnlAvailability: TPanel;
    cmbAvailableDates: TComboBox;
    btnQues223: TButton;
    Label4: TLabel;
    Panel2: TPanel;
    procedure btnQues221Click(Sender: TObject);
    Function determineDayTotal(sDateOfVisit: string): integer;
    procedure btnQues222Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure btnQues223Click(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;
const
  rCostPerPerson = 75.00;
  rTourGuide = 300.00;
var
  frmQuestion2: TfrmQuestion2;
  objExcursion: TExcursion;

implementation

{$R *.dfm}
//=====================================================================
//Question 2.2.1
//=====================================================================
procedure TfrmQuestion2.btnQues221Click(Sender: TObject);
var
  sSchoolName, sDate: string;
  iGroupsize: integer;
  bTourGuide: boolean;
begin
  sSchoolName := edtSchoolName.Text;
  sDate := lstVisitDate.Items[lstVisitDate.ItemIndex];
  iGroupSize := StrToInt(edtGroupSize.Text);
```

```
  if chbTourGuide.Checked then
    bTourGuide := true
  else
    bTourGuide := false;

  objExcursion := TExcursion.Create(sSchoolName, sDate, iGroupSize,
bTourGuide);
  ShowMessage('Excursion object has been instantiated.');
  pnlAvailability.Hide;
end;
//=====================================================================
//Question 2.2.2(a)
//=====================================================================
function TfrmQuestion2.determineDayTotal(sDateOfVisit: string): integer;
Var
  sSchoolName, sDate: string;
  iGroupSize, iTotal: integer;
  bTourGuide: boolean;
  txtFile: TextFile;
  sLine: string;
begin
  if not FileExists('DataQ2.txt') then
  begin
    MessageDlg('File does not exists.', mtError, [mbOk], 0);
    Exit;
  end;
  iTotal := 0;
  AssignFile(txtFile, 'DataQ2.txt');
  Reset(txtFile);

  while NOT EOF(txtFile) do
  begin
    readln(txtFile, sLine);
    sSchoolName := copy(sLine, 1, pos('#', sLine) - 1);
    Delete(sLine, 1, pos('#', sLine));
    sDate := copy(sLine, 1, pos('#', sLine) - 1);
    Delete(sLine, 1, pos('#', sLine));
    iGroupSize := StrToInt(sLine);
    if sDate = sDateOfVisit then
      iTotal := iTotal + iGroupSize;
  end; // while
  CloseFile(txtFile);
  Result := iTotal;
end;
//=====================================================================
//Question 2.2.2(b)
//=====================================================================
procedure TfrmQuestion2.btnQues222Click(Sender: TObject);
var
  I, iTotal: integer;
  sDate: string;
  bSuccessful: boolean;
begin
  bSuccessful:= false;
  iTotal := determineDayTotal(objExcursion.getDateOfVisit);
  if objExcursion.isConfirmed(iTotal) then
   begin
    ShowMessage(objExcursion.toString + #13 + 'Amount to be paid: ' +
FloatToStrF
        (objExcursion.calcAmount(rCostPerPerson,rTourGuide), ffCurrency, 8,
2));
    bSuccessful := true;
   end
```

```
  else
   begin
    ShowMessage('There is no space on the date selected.');
    cmbAvailableDates.Clear;
    pnlAvailability.Show;
    for I := 0 to 4 do
     begin
      sDate := lstVisitDate.Items[I];
      iTotal := determineDayTotal(sDate);
      if objExcursion.isConfirmed(iTotal) then
       begin
        cmbAvailableDates.Items.Add(sDate);
       end;
     end;
  end;
end;
//=====================================================================
//Question 2.2.3
//=====================================================================
procedure TfrmQuestion2.btnQues223Click(Sender: TObject);
var
  sDate : string;
  iIndex: integer;
begin
  if cmbAvailableDates.Items.Count > 0 then
  begin
    objExcursion.setVisitDate
      (cmbAvailableDates.Items[cmbAvailableDates.ItemIndex]);
    ShowMessage(objExcursion.toString + #13 + 'Amount to be paid: ' +
FloatToStrF
        (objExcursion.calcAmount(rCostPerPerson,rTourGuide), ffCurrency, 8,
2));
  end
  else
  begin
    ShowMessage('The application for ' + objExcursion.getSchoolName +
        ' is unsuccessful.');
  end;
end;
=======================================================================
// Provided code
=======================================================================
procedure TfrmQuestion2.FormActivate(Sender: TObject);
begin
  pnlAvailability.Hide;
  CurrencyString := 'R';
end;

end.
```

ANNEXURE I: SOLUTION FOR QUESTION 3: DELPHI

```
unit Question3_U;

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

type
  TfrmQuestion3 = class(TForm)
    Panel1: TPanel;
    GroupBox1: TGroupBox;
    btnTerminal1: TBitBtn;
    btnTerminal2: TBitBtn;
    btnTerminal3: TBitBtn;
    GroupBox2: TGroupBox;
    btnNorth: TBitBtn;
    btnSouth: TBitBtn;
    btnEast: TBitBtn;
    btnWest: TBitBtn;
    GroupBox3: TGroupBox;
    btnQues31: TButton;
    btnQues32: TButton;
    btnQues33: TButton;
    btnQues34: TButton;
    redQ3: TRichEdit;
    cmbQues3: TComboBox;
    GroupBox4: TGroupBox;
    procedure btnQues31Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure btnQues32Click(Sender: TObject);
    procedure btnTerminal1Click(Sender: TObject);
    procedure btnTerminal2Click(Sender: TObject);
    procedure btnTerminal3Click(Sender: TObject);
    procedure btnNorthClick(Sender: TObject);
    procedure btnSouthClick(Sender: TObject);
    procedure btnEastClick(Sender: TObject);
    procedure btnWestClick(Sender: TObject);
    procedure btnQues33Click(Sender: TObject);
    procedure btnQues34Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
//=====================================================================
// Provided code
//=====================================================================
var
  frmQuestion3: TfrmQuestion3;
  arrCodes: array [1 .. 10] of char = (
    'W',
    'A',
    'S',
    'R',
    'X',
    'D',
    'H',
    'P',
    'T',
    'L'
  );
```

```
  arrActivities: array [1 .. 10] of String = (
    'Water park',
    'Aquarium',
    'Sea',
    'Restaurants',
    'Shopping',
    'Diving',
    'Help desk',
    'Penguin park',
    'Shark tank',
    'Dolphin shows'
  );
  arrActCodes: array [1 .. 3, 1 .. 4] of String = (('DXWAT', 'HRDST',
      'STWLP', 'RDT'), ('SWA', 'SRXD', 'LWXH', 'SHA'),
    ('WLSR', 'AT', 'DATX', 'HW'));

  iTerminal: integer = 1;
  iDirection: integer = 1;
//=====================================================================
// Global variables
//=====================================================================
  arrDirections: array [1 .. 4] of string = (
    'North',
    'South',
    'East',
    'West'
  );
//=====================================================================
// Provided code
//=====================================================================
implementation

{$R *.dfm}
procedure TfrmQuestion3.btnTerminal1Click(Sender: TObject);
begin
  iTerminal := 1;
end;

procedure TfrmQuestion3.btnTerminal2Click(Sender: TObject);
begin
  iTerminal := 2;
end;

procedure TfrmQuestion3.btnTerminal3Click(Sender: TObject);
begin
  iTerminal := 3;
end;

procedure TfrmQuestion3.btnNorthClick(Sender: TObject);
begin
  iDirection := 1;
end;

procedure TfrmQuestion3.btnSouthClick(Sender: TObject);
begin
  iDirection := 2;
end;

procedure TfrmQuestion3.btnEastClick(Sender: TObject);
begin
  iDirection := 3;
end;
```

```
procedure TfrmQuestion3.btnWestClick(Sender: TObject);
begin
  iDirection := 4;
end;
//=====================================================================
//Question 3.1
//=====================================================================
procedure TfrmQuestion3.btnQues31Click(Sender: TObject);
var
  iRow, iCol: integer;
  sLine: string;
begin
  redQ3.Lines.Add('' + #9 + 'North' + #9 + 'South' + #9 +
      'East' + #9 + 'West');
  for iRow := 1 to 3 do
  begin
    sLine := 'Terminal ' + IntToStr(iRow) + #9;
    for iCol := 1 to 4 do
    begin
      sLine := sLine + arrActCodes[iRow, iCol] + #9;
    end;
    redQ3.Lines.Add(sLine);
  end;
end;

procedure TfrmQuestion3.FormActivate(Sender: TObject);
begin
  redQ3.Paragraph.TabCount := 4;
  redQ3.Paragraph.Tab[0] := 80;
  redQ3.Paragraph.Tab[1] := 130;
  redQ3.Paragraph.Tab[2] := 180;
  redQ3.Paragraph.Tab[3] := 230;
end;
//=====================================================================
//Question 3.2
//=====================================================================
procedure TfrmQuestion3.btnQues32Click(Sender: TObject);
var
  i, j: integer;
  sCodes: string;
begin
  redQ3.Clear;
  redQ3.Lines.Add('Terminal ' + IntToStr(iTerminal) + ', ' + arrDirections
      [iDirection]);
  sCodes := arrActCodes[iTerminal, iDirection];
  for i := 1 to length(sCodes) do
  begin
    for j := 1 to length(arrCodes) do
    begin
      if sCodes[i] = arrCodes[j] then
        redQ3.Lines.Add(arrActivities[j]);
    end;
  end;
end;
```

```
//=====================================================================
//Question 3.3
//=====================================================================
procedure TfrmQuestion3.btnQues33Click(Sender: TObject);
var
  iRow, iCol, iCount, iIndex: integer;
begin
  redQ3.Clear;
  iCount := 0;
  iIndex := cmbQues3.ItemIndex;
  redQ3.Lines.Add('Access routes to ' + cmbQues3.Items[cmbQues3.ItemIndex]
    );
  for iRow := 1 to 3 do
    for iCol := 1 to 4 do
    begin
      if pos(arrCodes[iIndex + 1], arrActCodes[iRow, iCol]) > 0 then
      begin
        redQ3.Lines.Add('Terminal ' + IntToStr(iRow) + ',  ' + arrDirections
            [iCol]);
        Inc(iCount);
      end;
    end;
  redQ3.Lines.Add(#13 + 'Number of access routes: ' + IntToStr(iCount));
end;
//=====================================================================
//Question 3.4
//=====================================================================
procedure TfrmQuestion3.btnQues34Click(Sender: TObject);
var
  iIndex, iRow, iCol: integer;

begin
  redQ3.Clear;
  redQ3.Lines.Add('Updated information:');
  redQ3.Lines.Add('');
  iIndex := cmbQues3.ItemIndex;
  for iRow := 1 to 3 do
    for iCol := 1 to 4 do
      if pos(arrCodes[iIndex + 1], arrActCodes[iRow, iCol]) > 0 then
      begin
        Delete(arrActCodes[iRow, iCol], pos(arrCodes[iIndex + 1],
            arrActCodes[iRow, iCol]), 1);
      end;

  ShowMessage('The access routes to ' + arrActivities[iIndex + 1]
      + ' are closed.');
 btnQues31.Click;
end;


end.
```