

SA's Leading Past Year

Exam Paper Portal



You have Downloaded, yet Another Great
Resource to assist you with your Studies 😊

Thank You for Supporting SA Exam Papers

Your Leading Past Year Exam Paper Resource Portal

Visit us @ www.saexampapers.co.za



**SA EXAM
PAPERS**



basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE12

INFORMATION TECHNOLOGY P1

NOVEMBER 2015

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 standardisation meeting to ensure that the guidelines are consistently interpreted and applied in the marking of candidates' work.
- Note that candidates 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 answer/solution, unless the specific instructions in the paper was not followed or the requirements of the question was not met.
- **Annexures A, B and C** (pages 3–9) include the marking grid for each question for using either one of the two programming languages.
- **Annexures D, E and F** (pages 10–19) contain examples of solutions for Java for Questions 1 to 3 in programming code.
- **Annexures G, H and I** (pages 20–31) contain examples of solutions for Delphi for Questions 1 to 3 in programming code.
- Copies of **Annexures A, B and C** (pages 3–9) should be made for each candidate 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	CANDIDATE'S MARKS
	<i>If a learner has a problem reading from a component, penalise only once for the error.</i>		
1.1	Button –[Question 1.1] Extract the start weight and height from the text box ✓ and convert both to real/double number ✓ Calculate BMI using the correct formula ✓ Display BMI in the output area ✓ formatted to 5 decimals ✓ If ✓ statements with correct ranges making provision for all three categories ✓ (<18.5) (>=18.5 to <=25) and (>25) Correct messages displayed ✓	8	
1.2	Button –[Question 1.2] Create a counter and initialise the counter ✓ Extract the goal weight from the text box and convert to number Check if start weight > goal weight ✓ Loop ✓ Increment counter ✓ Decrease weight by 0.375 ✓ Display counter <i>and weight</i> ✓ (no marks for formatting) else Display the message "Invalid value entered" ✓	7	

1.3	<p>Button –[Question 1.3]</p> <p>Extract the name from text box Convert name to uppercase ✓ Initialise an empty string for membership code ✓ <i>Remove vowels and spaces:</i> Loop through the name ✓ Check if not a vowel ✓ or space ✓ Join character to the membership code ✓</p> <p>Check if female and join -F- to membership code ✓ Check if male and join -M- to membership code ✓</p> <p>Generate random number in range 1 to 9 ✓ (1 and 9 included) Get the number of characters in membership code ✓ Calculate last two digits ✓ (the random number+10+length) Join to membership code ✓ (the random number + last 2 digits) Check if allergy check box is selected then join the * to the membership code ✓ Display the membership code in the text box ✓</p>	14	
1.4	<p>Button – [Question 1.4]</p> <p>Randomly select a number in the correct range ✓ (must include 20 members) Check if the selected entry in the array is a male ✓ or female ✓</p> <p>Use a conditional loop ✓ Randomly select the second number ✓ in the correct range Validate second number (if) ✓ to ensure the gender entry in the array is not the same as the first – use correct variables ✓ (<i>as loop condition</i>)</p> <p><i>Display in output area:</i> the first member ✓ and the second member selected ✓</p>	9	

1.5	Button – [Question 1.5] <i>Sort the array alphabetically:</i> Use two loops ✓ with valid counter values ✓ Compare membership codes ✓ (correct indexes) ✓ Swap the two values ✓✓✓ (-1 for each error to a maximum of 3) <i>Display members with allergies:</i> Use a loop ✓ Check if member has allergy ✓ Display the membership number ✓ <i>Display members without allergies:</i> Use a loop ✓ Check if member does not have an allergy ✓ Display the membership number	12	
	TOTAL:	50	

ANNEXURE B**SECTION B****QUESTION 2: MARKING GRID– OBJECT-ORIENTED PROGRAMMING**

CENTRE NUMBER:		EXAMINATION NUMBER:	
QUESTION	DESCRIPTION	MAX. MARKS	CANDIDATE'S MARKS
2.1.1	determineExpDate method: Extract the year value ✓ from registration date parameter ✓ and increment with a value of two ✓ Correctly combine with rest of date ✓	4	
2.1.2	Constructor: Definition with three correct parameters and data types ✓ Assign parameter values to the name ✓ and registration code attributes ✓ Use the determineExpDate method with argument to set the expiry date attribute ✓ Set the sessionsCompleted attribute to 0 ✓	5	
2.1.3	setSessionsCompleted Method definition with parameter ✓ Assign the parameter value to the sessionsCompleted method ✓	2	
2.1.4	increaseSessionsCompleted method: Method definition ✓ Increase sessionsCompleted attribute by 1 ✓	2	
2.1.5	evaluateProgress method: Method definition and correct parameter ✓ Calculate the percentage Use the attribute sessions completed value ✓ Use of parameter value (total session) ✓ Division ✓ (<i>Penalise integer division</i>) Check if percentage greater than 75 ✓ Return the message that the trainee qualifies as an instructor ✓ Else Return the percentage completed with % sign at the end ✓ formatted to two decimal places ✓	8	
2.1.6	toString method: Method definition ✓ Format on three different lines ✓ correct attributes ✓ (-1 for each incorrect attribute)(ignore []) Correct return statement ✓	4	

2.2.1	Button – [Question2.2.1] Instantiate a new trainee object ✓ with the correct order and data types of arguments ✓ Display the object using the toString method ✓	3	
2.2.2	Button – [Question2.2.2] Test if text file exists ✓ using if/try...except/try...catch <i>If the text file does not exist:</i> Display message ✓ and exit/close the program ✓ (or if.. then... else is constructed in such a way as to leave the procedure) <i>If the text file exists:</i> Enable buttons btn223 and btn224 ✓ Display the name of trainee in output area ✓ Open the text file to read from file: ✓✓ Delphi: AssignFile, RESET Java: Create object to read from file Call setSessionsCompleted method with 0 as a parameter ✓ Loop through file ✓ Read one line from text file ✓ Splitting the line to extract the trainee code ✓ (or test if the line contains the trainee code) Test if code matches the code of trainee selected ✓ Extract date ✓ If session has been completed ✓ Call the increaseSessions method ✓ Display the date in output area ✓ Display the object using the toString method NOTE: There are 2 ways to determine the sessions completed value for 2 marks. Method 1: Local variable used to count needs 3 steps: - <ul style="list-style-type: none"> • set variable to 0 • increment the variable inside the loop • call setSessionsCompleted method to set the value. Method 2: Using attributes of the object needs 2 steps:- <ul style="list-style-type: none"> • Use the setSessionsCompleted to set to 0 • Call the increaseSessions inside the loop to increment the attribute value. 	16	

2.2.3	Button – [Question 2.2.3] Get the date from the provided textbox ✓ (YYYY/MM/DD) Determine if check box is selected ✓ <i>Compile a string:</i> Registration code; date and "Completed" ✓ Call the method to increase sessions completed ✓ Else Registration code; date and "Not completed" ✓ <i>Write compiled string to text file:</i> <i>Open the text file to ADD to file: ✓</i> Delphi: APPEND Java: Create FileWriter object to append to file Write the string to the file writeln(Delphi)/println(Java) ✓ Close the file ✓ Display message that data was saved to file ✓ <i>toString</i> method to display information ✓	10	
2.2.4	Button – [Question 2.2.4] Extract the number of total sessions required from the textbox provided and convert to number ✓ Use of evaluateProgress method with argument ✓ Display the output of the progress ✓	3	
	TOTAL:	57	

ANNEXURE C**SECTION C****QUESTION 3: MARKING GRID–PROBLEM-SOLVING PROGRAMMING**

CENTRE NUMBER:		EXAMINATION NUMBER:	
QUESTION	DESCRIPTION	MAX. MARKS	CANDIDATE'S MARKS
Components	<i>Suitable components:</i> <ul style="list-style-type: none"> Processing: at least 2 buttons; ✓ Output: richedit/stringgrid/textarea ✓ 	2	
Program techniques	<i>Modular design:</i> ✓ <ul style="list-style-type: none"> Define and create at least one method/function/procedure and use the method/function/procedure correctly <i>Programming techniques:</i> (Any ONE of) ✓ <ul style="list-style-type: none"> Use of proper indentation Use of descriptive variable names Make use of inline comments Use appropriate data structures 	2	
Display arrays with headings in columns	Display column headings (Day 1 to Day 4) ✓ Display row headings (workshop topics) ✓ Outer loop (Number of workshops) ✓ with counter ✓ Inner loop (Number of days) ✓ Display 2d array values ✓ Correct formatting of columns ✓ Each workshop on new line ✓ NOTE: Any other method that generates the correct output without using a loop.	8	
Make a Booking	Get row index (<i>workshop</i>) ✓✓ and column index (<i>day</i>) ✓✓ from input Check in 2D array at selected index ✓ for available space (<20) ✓ <i>If space available:</i> Increment value in 2D array by one ✓ Call display method/button/write code to display ✓ Show message ✓ that booking is made including day and workshop ✓ <i>No space available:</i> Display message that workshop is fully booked ✓	11	

Full cases of water	Initialise bottles of water for each day to zero } Initialise total bottles of water to zero } ✓ Correct loops(for row and column) ✓ <i>Outer loop:4, Inner loop:6</i> Increment day totals with array value ✓ Increment total value ✓ Display the day and totals ✓ in columns ✓ Display the total number of water bottles ✓ Calculate number of cases of water by dividing by 24 ✓ Correctly rounded up ✓ Display the number of cases ✓	10	
Cancel a workshop	Get index of workshop to be cancelled, from input ✓ Loop ✓ using a variable for upper bound ✓ Remove workshop from workshop array ✓✓ Remove applicable values from 2D-array ✓✓ Decrease the counter for maximum number of workshops ✓ NOTE: Can also be done using nested loops. NOTE: If a flag is used: <ul style="list-style-type: none"> • Flagging ✓✓ the correct row ✓ • Provide code in the display to accommodate the flag ✓✓ • Provide code in the water bottle count to accommodate flag ✓✓ Remove the workshop from the combobox ✓ Display the updated array ✓ NOTE: If original display is called, it must accommodate the changed array/flagging.	10	
	TOTAL:	43	

SUMMARY OF CANDIDATE'S MARKS:

	SECTION A	SECTION B	SECTION C	
	QUESTION 1	QUESTION 2	QUESTION 3	GRAND TOTAL
MAX. MARKS	50	57	43	150
CANDIDATE'S MARKS				

ANNEXURE D: SOLUTION FOR QUESTION 1: JAVA

=====

Supplied code

```
=====
public class Question1_Memo extends javax.swing.JFrame {

    String[] arrMemberCodes = new String[20];

    public void fillMemberCodes() {
        arrMemberCodes[0] = "PRTHNMM-M-421";
        arrMemberCodes[1] = "LYYHNBB-F-623*";
        arrMemberCodes[2] = "DFGQWJJK-M-220*";
        arrMemberCodes[3] = "NBVGTYF-F-926";
        arrMemberCodes[4] = "NBGTRFSSD-F-322*";
        arrMemberCodes[5] = "NJKYTRRTG-M-928";
        arrMemberCodes[6] = "JBHGTYGFTF-F-121";
        arrMemberCodes[7] = "HGTJRJJ-F-522*";
        arrMemberCodes[8] = "KJHYTGFDWRQ-M-830";
        arrMemberCodes[9] = "NHYTRFDDD-M-221*";
        arrMemberCodes[10] = "NBVGTYGFG-M-424";
        arrMemberCodes[11] = "CVBGRXXS-M-726";
        arrMemberCodes[12] = "PLIUHYGTRF-M-323";
        arrMemberCodes[13] = "QWDFGENBG-M-423*";
        arrMemberCodes[14] = "RBRTHNDRKS-F-525";
        arrMemberCodes[15] = "MKJHTGFDD-M-625";
        arrMemberCodes[16] = "SDWRQWDDG-F-726";
        arrMemberCodes[17] = "HNGBBVFFDCCS-F-931";
        arrMemberCodes[18] = "NMBGHFDRLP-F-121";
        arrMemberCodes[19] = "BVCZZXGFDJK-M-122";
    }

    public Question1_Memo() {
        initComponents();
        this.setLocationRelativeTo(this);
        fillMemberCodes();
    }

    // Question 1.1
    private void btnQues1_1ActionPerformed(java.awt.event.ActionEvent evt)
    {
        double startWeight = Double.parseDouble(txfWeight.getText());
        double height = Double.parseDouble(txfHeight.getText());
        double bmi = startWeight / (height * height);
        String sBmi = String.format("%8.5f",bmi);
        txaOutput_1_1.setText("BMI = " + sBmi + "\n");
        if (bmi < 18.5) {
            txaOutput_1_1.append("Underweight");
        } else if (bmi <= 25) {
            txaOutput_1_1.append("Normal weight");
        } else {
            txaOutput_1_1.append("Overweight");
        }
    }
}
=====
```

```
=====
// Question 1.2
=====
private void btnQues1_2ActionPerformed(java.awt.event.ActionEvent evt)
{
    int numDays = 0;
    double startWeight = Double.parseDouble(txfWeight.getText());
    double goalWeight = Double.parseDouble(txfGoalWeight.getText());
    if (startWeight > goalWeight) {
        txaOutput_1_2.setText("Day\tWeight\n");
        while (goalWeight < startWeight) {
            numDays++;
            startWeight -= 0.375;
            txaOutput_1_2.append(numDays + "\t" +
                                String.format("%6.3f", startWeight) + "\n");
        }
    } else {
        txaOutput_1_2.setText("Invalid value entered");
    }
}
=====
// Question 1.3
=====
private void btnQues1_3ActionPerformed(java.awt.event.ActionEvent evt)
{
    String name = txfName.getText().toUpperCase();
    //OR membershipCode = membershipCode.replaceAll("[AEIOU ]", "");

    String membershipCode = "";
    for (int i = 0; i < name.length(); i++) {
        if (name.charAt(i) != 'A' && name.charAt(i) != 'E' &&
            name.charAt(i) != 'I'
            && name.charAt(i) != 'O' && name.charAt(i) != 'U' &&
            name.charAt(i) != ' ') {
            membershipCode += name.charAt(i);
        }
    }
    int numChar = membershipCode.length();

    if (rbtnFemale.isSelected()) {
        membershipCode += "-F-";
    }
    if (rbtnMale.isSelected()) {
        membershipCode += "-M-";
    }

    int randNum = (int)(Math.random() * 9) + 1;
    membershipCode = membershipCode + randNum +
        (randNum + 10 + numChar);

    if (chbAllergy.isSelected()) {
        membershipCode += '*';
    }

    txfMembershipNumber.setText(membershipCode);
}
=====
```

```
=====
// Question 1.4
=====
private void btnQues1_4ActionPerformed(java.awt.event.ActionEvent evt)
{
    int randomNumber1 = (int) (Math.random() * 20);
    String gender = "-M-";
    if (arrMemberCodes[randomNumber1].contains("-F-")) {
        gender = "-F-";
    }

    int randomNumber2;

    do {
        randomNumber2 = (int) (Math.random() * 20);
    } while (arrMemberCodes[randomNumber2].contains(gender));

    txtaOutput_1_4.setText("Premium members\n");
    txtaOutput_1_4.append("\n" + arrMemberCodes[randomNumber1]);
    txtaOutput_1_4.append("\n" + arrMemberCodes[randomNumber2]);
}
=====
// Question 1.5
=====
private void btnQues1_5ActionPerformed(java.awt.event.ActionEvent evt)
{
    for (int i = 0; i < 19; i++) {
        for (int j = i + 1; j < 20; j++) {

            if ((arrMemberCodes[i]).compareTo(arrMemberCodes[j]) > 0) {
                String temp = arrMemberCodes[i];
                arrMemberCodes[i] = arrMemberCodes[j];
                arrMemberCodes[j] = temp;
            }
        }
    }
    for (int i = 0; i < 20; i++) {
        if (arrMemberCodes[i].contains("*")) {
            txtaOutput_1_5.append(arrMemberCodes[i] + "\n");
        }
    }
    for (int i = 0; i < 20; i++) {

        if (!arrMemberCodes[i].contains("*")) {
            txtaOutput_1_5.append(arrMemberCodes[i] + "\n");
        }
    }
}
}
```

ANNEXURE E: SOLUTION FOR QUESTION 2: JAVA

=====

Supplied code

=====

```
public class Student {
    private String name;
    private String regCode;
    private String expiryDate;
    private int sessionsCompleted;
```

=====

```
// Question 2.1.1
```

=====

```
private String determineExpDate(String regDate) {
    int year = Integer.parseInt(regDate.substring(0, 4));
    year = year + 2;
```

```
    String expDate = year + regDate.substring(4);
    return expDate;
}
```

=====

```
// Question 2.1.2
```

=====

```
public Student(String name, String regCode, String regDate) {
    this.name = name;
    this.regCode = regCode;
    expiryDate = determineExpDate(regDate);
    sessionsCompleted = 0;
}
```

=====

```
// Question 2.1.3
```

=====

```
public void setSessionsCompleted(int counter) {
    sessionsCompleted = counter;
}
```

=====

```
// Question 2.1.4
```

=====

```
public void increaseSessionsCompleted() {
    sessionsCompleted++;
}
```

=====

```
// Question 2.1.5
```

=====

```
public String evaluateProgress(int total) {
    double percent = (sessionsCompleted / (double) total) * 100;
    if (percent > 75) {
        return (name + " qualifies as an instructor");
    } else {
        return ("Percentage completed: " + String.format("%-2.2f",
            percent) + "%");
    }
}
```

```
=====
// Question 2.1.6
=====
public String toString() {
return (name + " [" + regCode + "]\n" + "Expiry Date: " + expiryDate
      + "\nCompleted sessions: " + sessionsCompleted);
}

// Supplied code

public String getName() {
return name;
}

public String getCode() {
return regCode;
}

public String getExpDate() {
return expiryDate;
}

public int getSessionsCompleted() {
return sessionsCompleted;
}
}
```


GUI CLASS: QUESTION2_SOLUTION

=====

Supplied code

=====

Student objStudent;

=====

// Question 2.2.1

=====

```
private void btnQuestion_2_2_1ActionPerformed(java.awt.event.ActionEvent evt)
{
    objStudent = new Student(txfStudent.getText(), txfRegCode.getText(),
                             txfRegDate.getText());
    txaOutput.setText(objStudent.toString());
}
```

=====

// Question 2.2.2

=====

```
private void btnQuestion_2_2_2ActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        txaOutput.setText("Name of student: " + objStudent.getName() +
                           "\n");
        txaOutput.append("Dates of completed sessions:");
```

```
        Scanner inFile = new Scanner(new FileReader("DataQ2.txt"));
        objStudent.setSessionsCompleted(0);
        while (inFile.hasNext()) {
            String line = inFile.nextLine();
            Scanner scLine = new Scanner(line).useDelimiter("#");
            String codeDate = scLine.next();
            String code = codeDate.substring(0, 6);
            if (code.equals(objStudent.getCode())) {
                String date = codeDate.substring(codeDate.lastIndexOf(" "));
                String status = scLine.next();
                if (status.equalsIgnoreCase("Completed")) {
                    objStudent.increaseSessionsCompleted();
                    txaOutput.append("\n" + date);
                }
            }
        }
    }
```

```
    txaOutput.append("\n\n" + objStudent.toString());
```

```
    inFile.close();
```

```
    } catch (Exception e) {
        JOptionPane.showMessageDialog(null, "File does not exist");
        System.exit(0);
    }
```

```
    btnQ223.setEnabled(true);
```

```
    btnQ224.setEnabled(true);
```

```
}
```

```
=====
// Question 2.2.3
=====
private void btnQuestion_2_2_3ActionPerformed(java.awt.event.ActionEvent evt)
{
    String completed;
    if (chbCompleted.isSelected()) {
        completed = "Completed";
        objStudent.increaseSessionsCompleted();
    } else {
        completed = "Not completed";
    }
    String currentDate = txfTrainingDate.getText();
    String line = objStudent.getCode() + " trained on " + currentDate + "#"
        + completed;
    try {
        PrintWriter outFile = new PrintWriter(new FileWriter("DataQ2.txt",
            true));
        outFile.println(line);
        outFile.close();
        JOptionPane.showMessageDialog(null, "Information written to text
            file");
    } catch (Exception e) {
    }

    txaOutput.append("\n\n" + objStudent.toString());
}
=====
// Question 2.2.4
=====
private void
btnQuestion2_2_4ActionPerformed(java.awt.event.ActionEvent evt) {
    int totalSessions = Integer.parseInt(txfTotalSessions.getText());
    lblProgress.setText(objStudent.evaluateProgress(totalSessions));
}
```

ANNEXURE F: SOLUTION FOR QUESTION 3: JAVA

=====

Supplied code

```
=====
public class Question3_Memo extends javax.swing.JFrame {

    String[] arrWorkshops = {"Aerobics", "Bodybuilding", "Cardio",
                             "Dance", "Energy Supplements", "First Aid"};
    int[][] arrBookings = {{11, 14, 5, 14}, {15, 5, 20, 4},
                           {10, 14, 16, 20}, {20, 20, 20, 20}, {16, 7, 10, 7},
                           {10, 18, 13, 11}};

    // Declaration of global variable
    int numWorkshops = 6;

    // Display array with headings - call method to display
    private void btnDisplayActionPerformed(java.awt.event.ActionEvent evt) {
        display();
    }

    // Make a booking
    private void btnBookingActionPerformed(java.awt.event.ActionEvent evt) {
        String workshop = cmbWorkshops.getSelectedItem().toString();
        int day = lstDays.getSelectedIndex() + 1;
        String message = "";
        for (int r = 0; r < numWorkshops; r++) {
            if (workshop.equals(arrWorkshops[r])) {
                for (int c = 0; c < 4; c++) {
                    if (day == (c + 1)) {
                        if (arrBookings[r][c] < 20) {
                            arrBookings[r][c] = arrBookings[r][c] + 1;
                            display();
                            JOptionPane.showMessageDialog(null, workshop + " on
                                                                Day " + day + " is successfully booked");
                        }
                    }
                }
            }
        }
        JOptionPane.showMessageDialog(null, workshop + " on
                                        Day " + day + " is fully booked");
    }
}
=====
```

```
=====
// Calculate cases of bottled water
=====
private void btnWaterActionPerformed(java.awt.event.ActionEvent evt) {
    int bottles[] = new int[4];
    int totalBottles = 0;
    int cases = 0;
    for (int c = 0; c < 4; c++) {
        for (int r = 0; r < numWorkshops; r++) {
            bottles[c] = bottles[c] + arrBookings[r][c];
        }
        totalBottles = totalBottles + bottles[c];
    }
    txaOutput.setText("\nBottles of water needed:\n");
    for (int r = 0; r < 4; r++) {
        txaOutput.append(String.format("Day %-25s%-10s\n", (r + 1),
            bottles[r]));
    }
    double ans = totalBottles % 24;
    if (ans == 0) {
        cases = totalBottles / 24;
    } else {
        cases = (totalBottles / 24) + 1;
    }
    txaOutput.append(String.format("\n%-28s%-10s", "Total: ",
        totalBottles));

    txaOutput.append(String.format("\n%-28s%-10s", "Cases of bottled
        water needed: ", cases));
}
=====
// Cancel a workshop
=====
private void btnCancelActionPerformed(java.awt.event.ActionEvent evt) {
    int workshopNum = cmbWorkshops.getSelectedIndex();
    cmbWorkshops.removeItemAt(workshopNum);
    for (int i = workshopNum; i < numWorkshops - 1; i++) {
        arrWorkshops[i] = arrWorkshops[i + 1];
    }
    arrWorkshops[5] = "";
    for (int r = workshopNum; r < numWorkshops - 1; r++) {
        for (int c = 0; c < 4; c++) {
            arrBookings[r][c] = arrBookings[r + 1][c];
        }
    }
    numWorkshops--;
    display();
}
```

```
=====
// Method to display 2-d array with headings
=====
public void display() {
txaOutput.setText(String.format("%-25s", "Workshop"));
for (int i = 1; i <= 4; i++) {
txaOutput.append(String.format("%-10s", "Day " + i));
    }
txaOutput.append("\n\n");
for (int r = 0; r < numWorkshops; r++) {
txaOutput.append(String.format("%-25s", arrWorkshops[r]));
for (int c = 0; c < 4; c++) {
txaOutput.append(String.format("%-10s",
                                arrBookings[r][c]));
        }
txaOutput.append("\n");
    }
txaOutput.append("\n");
}
}
```

ANNEXURE G: SOLUTION FOR QUESTION 1: DELPHI

```
unit Question1U_Memo;

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

type
    TfrmQuestionONE = class(TForm)
    bmbClose: TBitBtn;
    lblFormHeading: TLabel;
        grbQuest11: TGroupBox;
        grbQuest12: TGroupBox;
        grbQuest13: TGroupBox;
        grbQuest14: TGroupBox;
    lblHWeight: TLabel;
    lblHeight: TLabel;
        btnQuestion1_1: TButton;
        redQ11: TRichEdit;
    edtWeight: TEdit;
    edtHeight: TEdit;
        lblHWeight2: TLabel;
    edtGoalWeight: TEdit;
        btnQuestion1_2: TButton;
        redQ12: TRichEdit;
    lblHName: TLabel;
    edtName: TEdit;
    rgpGender: TRadioGroup;
    grbAllergy: TGroupBox;
    chkAllergy: TCheckBox;
    lblHCode: TLabel;
        btnQuestion1_3: TButton;
    edtMembershipCode: TEdit;
        btnQuestion1_4: TButton;
        redQ14: TRichEdit;
        grbQuest15: TGroupBox;
        btnQuestion1_5: TButton;
        redQ15: TRichEdit;
    procedure FormCreate(Sender: TObject);
    procedure btnQuestion1_1Click(Sender: TObject);
    procedure btnQuestion1_2Click(Sender: TObject);
    procedure btnQuestion1_3Click(Sender: TObject);
    procedure btnQuestion1_4Click(Sender: TObject);
    procedure btnQuestion1_5Click(Sender: TObject);
    procedure bmbCloseClick(Sender: TObject);
    private
        { Private declarations }
    public
        { Public declarations }
    end;
```

```
var
frmQuestionONE: TfrmQuestionONE;

implementation

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

var
arrMemberCodes: array [1 .. 20] of String;
// =====
// Question 1.1
// =====
procedure TfrmQuestionONE.bmbCloseClick(Sender: TObject);
begin
Application.Terminate;
end;

procedure TfrmQuestionONE.btnQuestion1_1Click(Sender: TObject);
var
rWeight, rHeight, rBMI: Real;
begin
rWeight := StrToFloat(edtWeight.Text);
rHeight := StrToFloat(edtHeight.Text);
rBMI := rWeight / sqr(rHeight);
    redQ11.Clear;
redQ11.Lines.Add('BMI = ' + FloatToStr(rBMI));
if (rBMI < 18.5) then
redQ11.Lines.Add('Underweight')
else if (rBMI <= 25) then
redQ11.Lines.Add('Normal weight')
else
redQ11.Lines.Add('Overweight');
end;

// =====
// Question 1.2
// =====
procedure TfrmQuestionONE.btnQuestion1_2Click(Sender: TObject);
var
rWeight, rGoalWeight: Real;
iNumdays: Integer;
begin
    redQ12.Clear;
rWeight := StrToFloat(edtWeight.Text);
rGoalWeight := StrToFloat(edtGoalWeight.Text);
iNumdays := 0;
if (rWeight > rGoalWeight) then
begin
redQ12.Lines.Add('Day' + #9 + 'Weight');
while (rWeight > rGoalWeight) do
begin
inc(iNumdays, 1);
rWeight := rWeight - 0.375;
redQ12.Lines.Add(IntToStr(iNumdays) + #9 + FloatToStrF(rWeight,
    ffFixed, 8, 3));
```

```
end; // while
end // if
else
redQ12.Lines.Add('Invalid value entered');
end;

// =====
// Question 1.3
// =====
procedure TfrmQuestionONE.btnQuestion1_3Click(Sender: TObject);
var
sMembershipCode, sName: String;
A, iRandom, iNumLetters, iSum: Integer;
sCheckNum : String;

begin
sName := Uppercase(edtName.Text);
sMembershipCode := '';
  For A := 1 to Length(sName) do
if NOT(sName[A] IN ['A', 'E', 'I', 'O', 'U', #32]) then
sMembershipCode := sMembershipCode + sName[A];
iNumLetters := length(sMembershipCode);
case rgpGender.ItemIndex of
  0: sMembershipCode := sMembershipCode + '-F-';
  1: sMembershipCode := sMembershipCode + '-M-';
end;

iRandom := Random(9) + 1;
iSum := iRandom + 10 + iNumLetters;
sCheckNum := IntToStr(iRandom) + IntToStr(iSum);
sMembershipCode := sMembershipCode + sCheckNum;
if chkAllergy.Checked then
sMembershipCode := sMembershipCode + '*';

edtMembershipCode.Text := sMembershipCode;
end;
// =====
// Question 1.4
// =====
procedure TfrmQuestionONE.btnQuestion1_4Click(Sender: TObject);
var
iFirst, iSecond : Integer;
sSeekGender      : String;
begin
iFirst := Random(20) + 1;
if pos('-M-', arrMemberCodes[iFirst]) = 0 then
sSeekGender := '-M-'
else
sSeekGender := '-F-';
repeat
iSecond := Random(20) + 1;
until (pos(sSeekGender, arrMemberCodes[iSecond]) > 0);
  redQ14.Clear;
redQ14.Lines.Add('Premium members' + #13);
redQ14.Lines.Add(arrMemberCodes[iFirst]);
redQ14.Lines.Add(arrMemberCodes[iSecond]);
end;
```



```
// =====  
// Question 1.5  
// =====  
procedure TfrmVraagEen.btnQuestion1_5Click(Sender: TObject);  
var  
i, j: Integer;  
temp: String;  
begin  
for i := 1 to 19 do  
for j := i + 1 to 20 do  
begin  
if arrMemberCodes[i] > arrMemberCodes[j] then  
begin  
temp := arrMemberCodes[i];  
arrMemberCodes[i] := arrMemberCodes[j];  
arrMemberCodes[j] := temp;  
end;  
end;  
for i := 1 to 20 do  
if pos('*', arrMemberCodes[i]) > 0  
then redQ15.Lines.Add(arrMemberCodes[i]);  
  
for i := 1 to 20 do  
if pos('*', arrMemberCodes[i]) = 0  
then redQ15.Lines.Add(arrMemberCodes[i]);  
end;  
// =====  
procedure TfrmQuestionONE.FormCreate(Sender: TObject);  
begin  
arrMemberCodes[1] := 'PRTHNMM-M-421';  
arrMemberCodes[2] := 'LYYHNBB-F-623*';  
arrMemberCodes[3] := 'DFGQWJJK-M-220*';  
arrMemberCodes[4] := 'NBVGTYF-F-926';  
arrMemberCodes[5] := 'NBGTRFSSD-F-322*';  
arrMemberCodes[6] := 'NJKYTRRTG-M-928';  
arrMemberCodes[7] := 'JBHGTYGFTF-F-121';  
arrMemberCodes[8] := 'HGTJRJJ-F-522*';  
arrMemberCodes[9] := 'KJHYTGFDWRQ-M-830';  
arrMemberCodes[10] := 'NHYTRFDDD-M-221*';  
arrMemberCodes[11] := 'NBVGTYGFG-M-424';  
arrMemberCodes[12] := 'CVBGRXXS-M-726';  
arrMemberCodes[13] := 'PLIUYHGTRF-M-323';  
arrMemberCodes[14] := 'QWDFGENBG-M-423*';  
arrMemberCodes[15] := 'RBRTHNDRKS-F-525';  
arrMemberCodes[16] := 'MKJHTGFDD-M-625';  
arrMemberCodes[17] := 'SDWRQWDDG-F-726';  
arrMemberCodes[18] := 'HNGBBVFFDCCS-F-931';  
arrMemberCodes[19] := 'NMBGHFDRLP-F-121';  
arrMemberCodes[20] := 'BVCZZXGFDJK-M-122';  
end;  
end.
```

ANNEXURE H: SOLUTION FOR QUESTION 2: DELPHI**OBJECT CLASS: STUDENT**

```
unit StudentU;

interface

type
    TStudent = class(TObject)
private
    FName           : String;
    fRegCode        : String;
    fExpiryDate     : String;
    fSessionsCompleted : Integer;

    function determineExpDate(sDate : String) : String;

public
    constructor Create(Name, RegCode, RegDate: String);
    procedure setSessionsCompleted(iSessions : Integer);
    procedure increaseSessionsCompleted;
    function evaluateProgress(iMax : Integer) : String;
    function toString : String;
    function GetName : String;
    function GetCode : String;
    function GetExpDate: String;
    function GetSessionsCompleted: Integer;
end;

implementation

uses SysUtils, Math;
{ TStudent }
//=====
//Question 2.1.1
//=====
function TStudent.determineExpDate(sDate: String): String;
var
    iYear :Integer;
begin
    iYear := StrToInt(copy(sDate,1,4)) + 2;
    result := IntToStr(iYear) + copy(sdate,5);
end;
//=====
//Question 2.1.2
//=====
constructor TStudent.Create(Name, RegCode, RegDate: String);
begin
    FName           := Name;
    fRegCode        := RegCode;
    fExpiryDate     := determineExpDate(RegDate);
    fSessionsCompleted := 0;
end;
```

```
//=====
//Question 2.1.3
//=====
procedure TStudent.setSessionsCompleted(iSessions : Integer);
begin
  fSessionsCompleted := iSessions;
end;

//=====
//Question 2.1.4
//=====
procedure TStudent.increaseSessionsCompleted;
begin
  Inc(fSessionsCompleted, 1);
end;

//=====
//Question 2.1.5
//=====
function TStudent.evaluateProgress(iMax: Integer): String;
var
  rProgress :Real;
begin
  rProgress := (fSessionsCompleted / iMax) * 100;
  if rProgress >= 75 then
    Result := fName + ' qualifies as an instructor'
  else
    Result := 'Percentage completed: ' +
      FloatToStrf(rProgress, ffFixed, 2, 2) + '%';
end;

//=====
//Question 2.1.6
//=====
function TStudent.toString: String;
begin
  Result := fName + ' [' + fRegCode + ']' + #13 +
    'Expiry date: ' + fExpiryDate + #13 +
    'Completed sessions: ' + IntToStr(fSessionsCompleted);
end;

//===== Code supplied =====
function TStudent.GetName: String;
begin
  Result := fName;
end;

function TStudent.GetCode: String;
begin
  Result := fRegCode;
end;

function TStudent.GetExpDate: String;
begin
  Result := fExpiryDate;
end;

function TStudent.GetSessionsCompleted: Integer;
begin
```

```
Result := fSessionsCompleted;
end;
```

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

MAIN FORM UNIT: QUESTION2_U.PAS

```
unit Question2U_Memo;

interface

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

type
  TfrmQuestionTWO = class(TForm)
  bmbClose: TBitBtn;
  lblFormHeading: TLabel;
    redQ2: TRichEdit;
  pnlButtons: TPanel;
    btnQuestion222: TButton;
    btnQuestion221: TButton;
    pnlQ223: TPanel;
    btnQuestion223: TButton;
    Label1: TLabel;
  edtTotalSessions: TEdit;
    btnQuestion224: TButton;
  pnlProgress: TPanel;
  lblProgress: TLabel;
  lblCompleted: TLabel;
  chkCompleted: TCheckBox;
  lblTrainingDate: TLabel;
  edtTrainingDate: TEdit;
  lblDate: TLabel;
  lblRegCode: TLabel;
  edtRegCode: TEdit;
  edtDate: TEdit;
  edtStudent: TEdit;
  procedure FormCreate(Sender: TObject);
  procedure btnQuestion221Click(Sender: TObject);
  procedure btnQuestion222Click(Sender: TObject);
  procedure btnQuestion223Click(Sender: TObject);
  procedure btnQuestion224Click(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmQuestionTWO: TfrmQuestionTWO;

implementation

var
  objStudent: TStudent;
{$R *.dfm}
{$R+}
```

```
//=====
//Question 2.2.1
//=====
procedure TfrmQuestionTWO.btnQuestion221Click(Sender: TObject);

begin
objStudent := TStudent.Create(edtStudent.text, edtRegCode.text,
                               edtDate.text);

    redQ2.Lines.Clear;
redQ2.Lines.Add(objStudent.toString);
end;

//=====
//Question 2.2.2
//=====
procedure TfrmQuestionTWO.btnQuestion222Click(Sender: TObject);
var
    TxtFile: Textfile;
sLine, sRegCode, sDate: String;
iCount: Integer;
begin

if not FileExists('DataQ2.txt') then
begin
MessageDlg('File does not exists.', mtError, [mbOk], 0);
    Exit;
end;

AssignFile(TxtFile, 'DataQ2.txt');
Reset(TxtFile);

iCount := 0;
    redQ2.Clear;
redQ2.Lines.Add('Name of student: ' + objStudent.GetName);
redQ2.Lines.Add('Dates of completed sessions:');
while NOT EOF(TxtFile) do
begin
readln(TxtFile, sLine);
if pos(objStudent.GetCode, sLine) = 1 then
begin
Delete(sLine, 1, pos('on ', sLine) + 2);
sDate := copy(sLine, 1, pos('#', sLine) - 1);
Delete(sLine, 1, pos('#', sLine));
if sLine = 'Completed' then
begin
redQ2.Lines.Add(sDate);
inc(iCount, 1);
end;
end;
end; // while
objStudent.setSessionsCompleted(iCount);
CloseFile(TxtFile);

redQ2.Lines.Add(#10);
redQ2.Lines.Add(objStudent.toString);

btnQuestion223.Enabled := true;
btnQuestion224.Enabled := true;
end;
```

```
//=====
//Question 2.2.3
//=====
procedure TfrmQuestionTWO.btnQuestion223Click(Sender: TObject);
var
    TxtFile: Textfile;
    sLine, sCompleted, sSesDate: String;
begin
    sSesDate := edtTrainingDate.text;

    if chkCompleted.Checked = false then
        sCompleted := 'Not Completed'
    else
        begin
            sCompleted := 'Completed';
            objStudent.increaseSessionsCompleted;
        end;

    sLine := objStudent.GetCode + ' trained on ' + sSesDate + '#' + sCompleted;

    AssignFile(TxtFile, 'DataQ2.txt');
    Append(TxtFile);
    writeln(TxtFile, sLine);
    CloseFile(TxtFile);

    ShowMessage('Information was successfully written to the file');

    redQ2.Lines.Clear;
    redQ2.Lines.Add(objStudent.toString);
end;

//=====
//Question 2.2.4
//=====
procedure TfrmQuestionTWO.btnQuestion224Click(Sender: TObject);
var
    iSessions: Integer;
    sProgress: String;
begin
    iSessions := StrToInt(edtTotalSessions.text);
    sProgress := objStudent.evaluateProgress(iSessions);
    lblProgress.Caption := sProgress;
end;

// =====
// Supplied code
// =====
procedure TfrmQuestionTWO.FormCreate(Sender: TObject);
begin
    btnQuestion223.Enabled := false;
    btnQuestion224.Enabled := false;
end;
end.
```

ANNEXURE I: SOLUTION FOR QUESTION 3: DELPHI

```
unit Question3U;

interface

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

type
  TfrmQuestion3 = class(TForm)
    pnlClose: TPanel;
    bmbClose: TBitBtn;
    pnlInput: TPanel;
    grpInput: TGroupBox;
    lblWorkshopQuestion: TLabel;
    lblTopic: TLabel;
    lblDay: TLabel;
    cboTopic: TComboBox;
    lblOutput: TLabel;
    lbluserComponents: TLabel;
    redDisplay: TRichEdit;
    btnDisplay: TButton;
    btnBook: TButton;
    btnCancelWorkshop: TButton;
    btnWater: TButton;
    lstDay: TListBox;
    Label1: TLabel;
    procedure btnDisplayClick(Sender: TObject);
    procedure btnBookClick(Sender: TObject);
    procedure display;
    procedure btnCancelWorkshopClick(Sender: TObject);
    procedure btnWaterClick(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmQuestion3: TfrmQuestion3;

implementation

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

CONST
  MaxRow: Integer = 6;
  MaxCol: Integer = 4;

var
  arrWorkshops: Array [1 .. 6] of String = (
    'Aerobics',
    'Bodybuilding',
    'Cardio',
    'Dancing',
    'Energy Supplements',
    'First Aid'
  );
```

```
numWorkshops: Integer = 6;

arrBookings: Array [1 .. 6, 1 .. 4] of Integer = ((11, 14, 5, 14),
    (15, 5, 20, 4), (10, 14, 16, 20), (20, 20, 20, 20), (16, 7, 10, 7),
    (10, 18, 13, 11));
//=====
// Display
//=====
procedure TfrmQuestion3.btnDisplayClick(Sender: TObject);
begin
display;
end;
//=====
// Make a booking
//=====
procedure TfrmQuestion3.btnBookClick(Sender: TObject);
var
sWorkshop, sMsg: String;
i, iDay, iWShop: Integer;
begin
sWorkshop := cboTopic.Text;
iWShop := 0;
for i := 1 to numWorkshops do
if sWorkshop = arrWorkshops[i] then
iWShop := i;
if iWShop > 0 then
begin
iDay := lstDay.ItemIndex + 1;
if arrBookings[iWShop, iDay] < 20 then
begin
Inc(arrBookings[iWShop, iDay]);
sMsg := sWorkshop + ' on Day ' + IntToStr(iDay) + ' is successfully
        booked';
end
else
sMsg := sWorkshop + ' on Day ' + IntToStr(iDay) + ' is fully booked';
end
else
sMsg := 'Workshop: ' + sWorkshop + ' Not available';
MessageDlg(sMsg, mtInformation, [mbOk], 0);
display;
end;
//=====
// Cancel a workshop
//=====
procedure TfrmQuestion3.btnCancelWorkshopClick(Sender: TObject);
var
i, iWShop, iDay, iRemoveLine: Integer;
sLine, sWorkshop: String;

begin
iRemoveLine := 0;
sWorkshop := cboTopic.Text;
for i := 1 to numWorkshops do
if sWorkshop = arrWorkshops[i] then
begin
iRemoveLine := i;
for iWShop := iRemoveLine to numWorkshops - 1 do
arrWorkshops[iWShop] := arrWorkshops[iWShop + 1];
```



```

for iDay := 1 to 4 do
for iWShop := iRemoveLine to numWorkshops - 1 do
arrBookings[iWShop, iDay] := arrBookings[iWShop + 1, iDay];

Dec(numWorkshops);
end;
display;
end;
//=====
// Determinethe number of cases of bottled water needed
//=====
procedure TfrmQuestion3.btnWaterClick(Sender: TObject);
var
iWShop, iDay, iTot, iDayTot: Integer;
sLine: String;

begin
redDisplay.Paragraph.TabCount := 4;
redDisplay.Paragraph.Tab[0] := 156;
redDisplay.Paragraph.Tab[1] := 200;
redDisplay.Paragraph.Tab[2] := 250;
redDisplay.Paragraph.Tab[3] := 300;
redDisplay.Lines.Add('Bottles of water needed:');
iTot := 0;
for iDay := 1 to 4 do
begin
iDayTot := 0;
for iWShop := 1 to numWorkshops do
iDayTot := iDayTot + arrBookings[iWShop, iDay];
redDisplay.Lines.Add('Day ' + IntToStr(iDay) + #9 + IntToStr(iDayTot));
iTot := iTot + iDayTot;
end;
redDisplay.Lines.Add(#10 + 'Total: ' + #9 + IntToStr(iTot));
redDisplay.Lines.Add('Cases of bottled water needed: ' +
FloatToStr(Round((iTot/24) + 0.5)));
end;
//=====
// Display
//=====
procedure TfrmQuestion3.display;
var
iWShop, iDay: Integer;
sLine: String;
begin
redDisplay.Clear;
redDisplay.Paragraph.TabCount := 4;
redDisplay.Paragraph.Tab[0] := 150;
redDisplay.Paragraph.Tab[1] := 200;
redDisplay.Paragraph.Tab[2] := 250;
redDisplay.Paragraph.Tab[3] := 300;
redDisplay.Lines.Add
('Workshop' + #9 + 'Day 1' + #9 + 'Day 2' + #9 + 'Day 3' + #9 + 'Day 4' +
#10);
for iWShop := 1 to numWorkshops do
begin
sLine := arrWorkshops[iWShop];
for iDay := 1 to 4 do
sLine := sLine + #9 + IntToStr(arrBookings[iWShop, iDay]);
redDisplay.Lines.Add(sLine);
end;
end;

end.

```