

# INFORMATION TECHNOLOGY P1

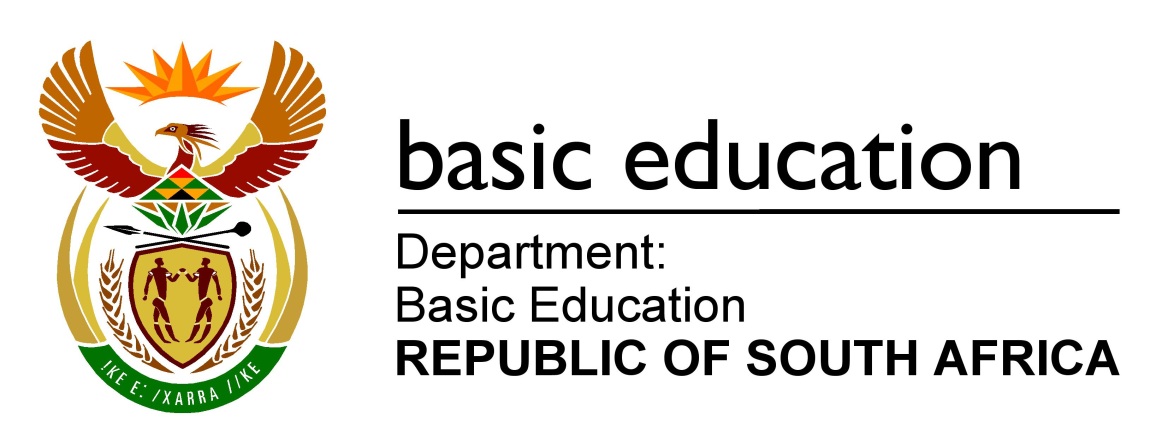
# EXEMPLAR 2018

# MARKING GUIDELINES

# NATIONAL

# SENIOR CERTIFICATE

# GRADE12



**MARKS: 150**

**These marking guidelines consist of 25 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 24) 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 Register Player]**  Test if name edit box > '' ✓and ✓e-mail edit box > ''✓  Display message which includes the name and e-mail address entered ✓  Else ✓  Display a suitable message – data incomplete ✓ | | **6** |  |
| 1.2 | **Button - [1.2 Calculate Cost of Meal]**  Variable declaration  Integer✓ Real✓ String✓  Test cbMealOptions.ItemIndex for selected meal✓  Use the case/if statements✓ to test for three options ✓  and assign correct price to variable (rMealPrice)✓  Obtain number of meals from spin edit✓  Calculate cost of meals (NumOfMeals \* rMealPrice)✓  Test if chbVegetarian is checked ✓  Calculate ✓and decrease cost amount by 10% ✓ (rMealsCost := rMealsCost \* 0.9)  Also accept any other correct formula  Display final cost in edit box as currency with 2 decimals✓  Change colour of font of cost edit box to clBlue✓ | | **14** |  |
| 1.3 | **Button - [1.3 Calculate Tables and Power]**  Variable declaration  All integer ✓  Obtain number of players from edit box and convert to int✓  Calculate number of tables required  (iNumTables := iNumPlayers DIV TableSize) ✓  if number of players not divisible by TableSize✓  Increase number of tables by 1✓  Calculate power required correctly ✓  Convert power required to String and display ✓  Convert number of tables to String and display ✓ | | **8** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 1.4 | **Button - [1.4 Calculate Days]**  Variable declaration  All integer ✓  Clear memo component✓  Obtain number of units from edit box, convert to integer✓  Generate random value✓ in correct range ✓as daily usage  Convert daily usage to String and display in edit box✓  Initialise loop counter to 0✓  While units >= daily usage do✓  Increment loop counter by 1✓  Subtract daily usage from units✓  Display loop counter ✓and units left, converted to strings✓ | **12** |  |
|  | **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 | **[Show All Players]**  SELECT \* ✓FROM tblPlayers✓ORDER BY Name✓ | | **3** |  |
| 2.1.2 | **[Players Born in September]**  SELECT Name, DateOfBirth FROM tblPlayers✓WHERE Month(DateOfBirth) ✓= 9✓ | | **3** |  |
| 2.1.3 | **[Average Game Times]**  SELECT GameDate, ✓Round(Avg(Duration),1) ✓AS AverageDuration✓FROM tblGames✓  GROUP BY GameDate✓ | | **5** |  |
| 2.1.4 | **[Highest Scores]**  SELECT Name, Max(Score) ✓AS HighestScore✓FROM tblPlayers P, tblGames G ✓WHERE P.PlayerID = G.PlayerID✓GROUP BY Name✓ | | **5** |  |
| 2.1.5 | **[Add Game]**  INSERT INTO tblGamesVALUES ✓ (76,#2017/12/24#,'HM008',250,6655)  All data items listed and correct values ✓  In correct order ✓correct format ✓ | | **4** |  |
|  | **Subtotal:** | | **20** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 2.2 | **DATABASE MANIPULATION** |  |  |
| 2.2.1 | **Button – [Domain Users Count]**  Obtain domain entered from InputBox✓  Display column headings✓  Initialise counter to 0  Set tblPlayers to start reading first record✓  Loop while NOT tblPlayers.Eof✓  Test if pos(sDomain,tblPlayers['Email']) > 0 ✓  Display name and email address from tblPlayers✓  Increment counter by 1✓  Go to next record in tblPlayers✓  End loop  Display message with domain and counter ✓  converted to String | **9** |  |
| 2.2.2 | **Button – [Change Score]**  Set tblGames to edit mode ✓  Add 250 points✓ to the content of selected Score field in tblGames✓  Post the change that was made ✓ | **4** |  |
| 2.2.3 | **Button – [Add Game]**  tblGames.Insert✓  tblGames['GameIndx'] :=tblGames.RecordCount + 1✓  tblGames['Gamedate'] := Date ✓  tblGames['PlayerID'] := tblPlayers['PlayerID'] ✓  tblGames['Duration'] := inputBox('Game Duration',  'Enter duration of game','') ✓  tblGames['Score'] := inputBox('Game Score',  'Enter score','') ✓  tblGames.Post✓ | **7** |  |
|  | **Subtotal:** | **20** |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **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 | Method heading ✓  result := fEmail ✓ | | **2** |  |
| 3.1.2 | Method heading ✓  fApproved :=✓Boolean parameter value ✓ | | **3** |  |
| 3.1.3 | Initialise Boolean variable to true✓  If the first character ✓ not uppercase letter ✓  Boolean variable = false ✓  result:= bPass ✓ | | **5** |  |
| 3.1.4 | fName:= name  if testEmail ✓ (email) ✓  fEmail:= email✓  else✓  fEmail:= 'ERROR' ✓  fDateOfBirth:= dateOfBirth  fApproved:= false | | **5** |  |
| 3.1.5 | Method heading:  Function name with correct data type✓ and parameter ✓  iRealAge:=YearsBetween(date,StrToDate(fDateOfBirth))✓✓  if iRealAge>= age ✓  result := 'APPROVED'✓  else  result := 'REJECTED'✓ | | **7** |  |
|  | **Subtotal:** | | **22** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 3.2.1 | **Button [Submit Details]**  objPlayer ✓:= TPlayer.create✓  (edtName.text,edtEmail.text,edtDateOfBirth.text) ✓✓  // Order and number of parameters  redDisplay.Clear✓ edtAgeResult.clear✓  redDisplay.Lines.Add(objPlayer.toString) ✓ | **7** |  |
| 3.2.2 | **Button [Test Age]**  edtAgeResult.Text ✓:=objPlayer.testAge✓  (StrToInt(edtAge.Text)✓) | **3** |  |
| 3.2.3 | **Button – [Approve]**  if (edtAgeResult.Text = 'APPROVED') ✓AND ✓  (objPlayer.getEmail<> 'ERROR') ✓  objPlayer.setApproved (true)✓  else  showMessage('The player''s age or e-mail has been rejected.') ✓  objPlayer.setApproved(false) ✓  redDisplay.Clear✓  redDisplay.Lines.Add(objPlayer.toString)✓ | **8** |  |
|  | **Subtotal:** | **18** |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **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 [Remove Duplicates]**  Initialise newCount to 0✓  Loop from 1 to iCount(outer loop) ✓ (i)  Set flag to false✓  Loop from 1 to newCount (inner loop) ✓ (j)  if arrNames[i] = arrTempNames[j] then✓  Set flag to true ✓  if arrScores[i] >arrTempScores[j] then✓  arrTempScores[j] := arrScores[i] ✓  end //inner loop  if NOT found then✓  Increment newCount by 1 ✓  arrTempNames[newCount] := arrNames[i] ✓  arrTempScores[newCount] := arrScores[i] ✓  end //if  end//outer loop  iCount := newCount  Loop from 1 to iCount✓  Begin loop  arrNames[i] := arrTempNames[i]✓  arrScores[i] := arrTempScores[i]✓  end loop  Call display procedure✓ | | **16** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **4.2** | **Button [Display Master Players]**  Initialise total as 0 ✓  Loop from 1 to iCount  iTot := iTot + arrScores[i] ✓  Calculate average (iAvg := Round(iTot / iCount)) ✓  Initialise iMastCount as 0✓  Loop from 1 to iCount✓  if arrScores[i] >iAvg then✓  Increment iMastCount by 1✓  arrMasters[iMastCount] :=  arrNames[i] + '#' + IntToStr(arrScores[i]) ✓  end if  end loop  Loop from 1 to iMastCount - 1 ✓  Loop from i + 1 to iMastCount✓  if arrMasters[i] >arrMasters[j] then✓  begin  sTemp := arrMasters[i]  arrMasters[i] := arrMasters[j] ✓ (swap)  arrMasters[j] := sTemp  end if  redDisplay.Lines.Add('MASTER PLAYERS');  redDisplay.Lines.Add('Scores above an average of ' + IntToStr(iAvg)) ✓  Loop from 1 to iMastCount  redDisplay.Lines.Add(arrMasters[i])✓ | **14** |  |
|  | **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**

unit Question1\_U;

interface

uses

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

type

TfrmQuestion1 = class(TForm)

Label7: TLabel;

GroupBox1: TGroupBox;

Label1: TLabel;

edtName: TEdit;

Label2: TLabel;

edtEmail: TEdit;

btnRegister: TButton;

GroupBox2: TGroupBox;

Label3: TLabel;

Label4: TLabel;

Label8: TLabel;

edtCost: TEdit;

sedNumMeals: TSpinEdit;

chbVegetarian: TCheckBox;

btnCalcCost: TButton;

GroupBox3: TGroupBox;

rbgTableSize: TRadioGroup;

btnCalcTables: TButton;

Label5: TLabel;

edtNumPlayers: TEdit;

Label6: TLabel;

Label9: TLabel;

edtNumTables: TEdit;

edtPower: TEdit;

GroupBox4: TGroupBox;

btnCalculate: TButton;

cbMealOptions: TComboBox;

Label12: TLabel;

edtUnits: TEdit;

Label13: TLabel;

edtDailyusage: TEdit;

memDisplay: TMemo;

procedure btnCalcTablesClick(Sender: TObject);

procedure btnRegisterClick(Sender: TObject);

procedure btnCalculateDaysClick(Sender: TObject);

procedure btnCalcCostClick(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

frmQuestion1: TfrmQuestion1;

implementation

{$R \*.dfm}

// =====================================================================

**// Question 1.1**

// =====================================================================

procedure TfrmQuestion1.btnRegisterClick(Sender: TObject);

begin

if (edtName.text > '') AND (edtEmail.Text > '') then

showMessage(edtName.text + ' with e-mail address ' + edtEmail.text +

' has been registered.')

else

showMessage('All the required data was not provided.');

end;

// =====================================================================

**// Question 1.2**

// =====================================================================

procedure TfrmQuestion1.btnCalcCostClick(Sender: TObject);

var

iNumOfMeals : Integer;

rMealPrice, rMealsCost : Real;

begin

case cbMealOptions.ItemIndex of

0: rMealPrice := 25;

1: rMealPrice := 32.80;

2: rMealPrice := 45.75;

end;

iNumOfMeals := sedNumMeals.Value;

rMealsCost := iNumOfMeals \* rMealPrice;

if chbVegetarian.checked then

rMealsCost := rMealsCost \* 1.1;

edtCost.Font.Color := clBlue;

edtCost.text := FloatToStrF(rMealsCost,ffCurrency,10,2);

end;

// =====================================================================

**// Question 1.3**

// =====================================================================

procedure TfrmQuestion1.btnCalcTablesPowerClick(Sender: TObject);

CONST

Tablesize = 6;

var

iNumPlayers, iNumTables, iPower: Integer;

begin

iNumPlayers := StrToInt(edtNumPlayers.text);

iNumTables := iNumPlayers DIV Tablesize;

if iNumPlayers MOD TableSize > 0 then

Inc(iNumTables);

iPower := Round(iNumPlayers \* 0.66);

edtPower.text := IntToStr(iPower);

edtNumTables.text := IntToStr(iNumTables);

end;

// =====================================================================

**// Question 1.4**

// =====================================================================

procedure TfrmQuestion1.btnCalculateDaysClick(Sender: TObject);

var

iUnits, iUsage, iCount: Integer;

begin

memDisplay.clear;

iUnits := StrToInt(edtUnits.text);

iUsage := Random(100) + 50;

edtDailyUsage.text := IntToStr(iusage);

iCount := 0;

while iUnits >= iUsage do

begin

Inc(iCount);

iUnits := iUnits - iUsage;

memDisplay.lines.add('Units after day ' + IntToStr(iCount) + ' : '

+ IntToStr(iUnits));

end;

end;

end.

**ANNEXURE F: SOLUTION FOR QUESTION 2**

unit question2\_U;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, DB, ADODB, StdCtrls, Grids, DBGrids, Mask, DBCtrls, ComCtrls, DateUtils, dbConnection\_U;

type

TfrmQuestion2 = class(TForm)

pgcQst2: TPageControl;

tsQst21: TTabSheet;

tsQst22: TTabSheet;

Label1: TLabel;

Label2: TLabel;

Label4: TLabel;

Label6: TLabel;

dbgPlayers: TDBGrid;

dbgGames: TDBGrid;

redDisplay: TRichEdit;

btnDomainUsersCount: TButton;

btnChangeScore: TButton;

btnAddRiver: TButton;

btnRestore: TButton;

Label3: TLabel;

Label5: TLabel;

dbgSQL: TDBGrid;

procedure connect;

procedure FormShow(Sender: TObject);

procedure btnDomainUsersCountClick(Sender: TObject);

procedure btnChangeScoreClick(Sender: TObject);

procedure btnAddRiverClick(Sender: TObject);

procedure btnRestoreClick(Sender: TObject);

procedure btnSQLqst211Click(Sender: TObject);

procedure btnSQLqst212Click(Sender: TObject);

procedure btnSQLqst213Click(Sender: TObject);

procedure btnSQLqst214Click(Sender: TObject);

procedure btnSQLbtn215Click(Sender: TObject);

end;

var

frmQuestion2: TfrmQuestion2;

// Databse will automatically connect

// These are the two tables that you work with

tblPlayers: TADOtable;

tblGames: TADOtable;

// DB SQL component

qryLanFans: TADOQuery;

sSQL : String;

implementation

{$R \*.dfm}

//======================================================================

// **Question 2.1.1**

//======================================================================

procedure TfrmQuestion2.btnSQLqst211Click(Sender: TObject);

begin

sSQL := 'SELECT \* FROM tblPlayers Order by Name';

qryLanFans.SQL.Clear;

qryLanFans.SQL.ADD(sSQL);

qryLanFans.Open;

end;

//======================================================================

// **Question 2.1.2**

//======================================================================

procedure TfrmQuestion2.btnSQLqst212Click(Sender: TObject);

begin

sSQL := 'SELECT Name, DateOfBirth FROM tblPlayers '+

'WHERE Month(DateOfBirth) = 9';

qryLanFans.SQL.Clear;

qryLanFans.SQL.ADD(sSQL);

qryLanFans.Open;

end;

//======================================================================

// **Question 2.1.3**

//======================================================================

procedure TfrmQuestion2.btnSQLqst213Click(Sender: TObject);

begin

sSQL :=('SELECT GameDate, Round(Avg(Duration),1) AS AverageDuration' +

' FROM tblGames GROUP BY GameDate');

qryLanFans.SQL.Clear;

qryLanFans.SQL.ADD(sSQL);

qryLanFans.Open;

end;

//======================================================================

// **Question 2.1.4**

//======================================================================

procedure TfrmQuestion2.btnSQLqst214Click(Sender: TObject);

begin

sSQL := 'SELECT Name, Max(Score) AS HighestScore ' +

'FROM tblPlayers P, tblGames G ' +

'WHERE P.PlayerID = G.PlayerID GROUP BY Name';

qryLanFans.SQL.Clear;

qryLanFans.SQL.ADD(sSQL);

qryLanFans.Open;

end;

//======================================================================

// **Question 2.1.5**

//======================================================================

procedure TfrmQuestion2.btnSQLbqst215Click(Sender: TObject);

begin

sSQL := 'INSERT INTO tblGames ' +

'VALUES (76,#2017/12/24#,''HM008'',250,6655)';

qryLanFans.SQL.Clear;

qryLanFans.SQL.ADD(sSQL);

qryLanFans.ExecSQL;

showMessage(IntToStr(qryLanFans.RowsAffected) + ' record/s added.');

end;

// =====================================================================

**// Question 2.2.1**

// =====================================================================

procedure TfrmQuestion2.btnDomainUsersCountClick(Sender: TObject);

var

sDomain: String;

iCount: Integer;

begin

// Provided code

iCount := 0;

redDisplay.Clear;

redDisplay.Paragraph.TabCount := 1;

redDisplay.Paragraph.Tab[0] := 90;

sDomain := Inputbox('Domain Email','Enter the e-mail domain to

search for','gmail.com');

redDisplay.lines.add('PLAYER' + #9 + 'E-MAIL ADDRESS');

tblPlayers.First;

while NOT tblPlayers.Eof do

begin

ifPos(sDomain,tblPlayers['Email']) > 0 then

begin

redDisplay.lines.add(tblPlayers['Name'] + #9 +

tblPlayers['Email']);

Inc(iCount);

end;

tblPlayers.Next;

end;

redDisplay.lines.add(#13 + 'TOTAL for ' + sDomain + ': '

+ IntToStr(iCount));

end;

// =====================================================================

**// Question 2.2.2**

// =====================================================================

procedure TfrmQuestion2.btnChangeScoreClick(Sender: TObject);

begin

tblGames.Edit;

tblGames['Score'] := tblGames['Score'] + 250;

tblGames.Post;

end;

// =====================================================================

**// Question 2.2.3**

// =====================================================================

procedure TfrmQuestion2.btnAddRiverClick(Sender: TObject);

begin

tblGames.Insert;

tblGames['GameIndx'] := tblGames.RecordCount + 1;

tblGames['Gamedate'] := Date;

tblGames['PlayerID'] := tblPlayers['PlayerId'];

tblGames['Duration'] := InputBox('Game Duration',

'Enter duration of game','');

tblGames['Score'] := InputBox('Game Score','Enter score','');

tblGames.Post;

end;

{$region Connect}

// Supplied Connection Code - DO NOT CHANGE.

// Database will automatically connect when run

var

dbLanFans: TConnection;

dsrPlayers: TDataSource;

dsrGames: TDataSource;

frmSQL: TfrmQst2SQL;

procedure TfrmQuestion2.FormShow(Sender: TObject);

begin

dbLanFans := TConnection.create;

dbLanFans.dbConnect;

connect;

pgcQst2.ActivePageIndex := 0;

end;

procedure TfrmQuestion2.connect;

begin

tblPlayers := TADOtable.Create(frmQuestion2);

tblPlayers.Connection := dbLanFans.dbConnection;

tblPlayers.TableName := 'tblPlayers';

tblPlayers.Open;

tblGames := TADOtable.Create(frmQuestion2);

tblGames.Connection := dbLanFans.dbConnection;

tblGames.TableName := 'tblGames';

tblGames.Open;

dsrPlayers := TdataSource.Create(frmQuestion2);

dsrPlayers.DataSet := tblPlayers;

dbgPlayers.DataSource := dsrPlayers;

dsrGames := TdataSource.Create(frmQuestion2);

dsrGames.DataSet := tblGames;

dbgGames.DataSource := dsrGames;

end;

**// Supplied Connection Code - DO NOT CHANGE.**

procedure TfrmQuestion2.btnRestoreClick(Sender: TObject);

var

bFail: boolean;

begin

dbLanFans.dbConnection.close;

tblPlayers.Destroy;

tblGames.Destroy;

DeleteFile('LanFans.mdb');

CopyFile('LanFansBackup.mdb','LanFans.mdb',bFail);

dbLanFans.dbConnect;

connect;

end;

procedure TfrmQuestion2.Button1Click(Sender: TObject);

begin

frmSQL := TfrmQst2SQL.Create(frmQuestion2);

frmSQL.ShowModal;

end;

{$endregion}

end.

**ANNEXURE G: SOLUTION FOR QUESTION 3**

**Object class**

unitplayer\_U;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,Dialogs, DateUtils;

type

TPlayer = class(TObject)

private

fEmail: String;

fName: String;

fDateOfBirth: String;

fApproved: Boolean;

functiontestEmail(email: String): Boolean;

public

constructor create(name, email, dateOfBirth: String);

functiontoString: String;

functiontestAge(age : Integer): String;

proceduresetApproved(approve :Bboolean);

functiongetEmail: String;

end;

implementation

{ TPlayer }

// =====================================================================

**// Question 3.1.1**

// =====================================================================

function TPlayer.getEmail: String;

begin

result := fEmail;

end;

// =====================================================================

**// Question 3.1.2**

// =====================================================================

procedure TPlayer.setApproved(approve: Boolean);

begin

fApproved := approve;

end;

// =====================================================================

**// Question 3.1.3**

// =====================================================================

function TPlayer.testEmail(email: String): Boolean;

var

iPosAt, iPosDot: Integer;

bPass: Boolean;

begin

bPass := true;

if NOT (Uppercase(email)[1] IN ['A'..'Z']) then

bPass := False;

result := bPass;

end;

// =====================================================================

**// Question 3.1.4**

// =====================================================================

constructorTPlayer.create(name, email, dateOfBirth: String);

begin

fName := name;

iftestEmail(email) then

fEmail := email

else

fEmail := 'ERROR';

fDateOfBirth := dateOfBirth;

fApproved := false;

end;

// =====================================================================

**// Question 3.1.5**

// =====================================================================

functionTPlayer.testAge(age: Integer): String;

var

iRealAge : Integer;

begin

iRealAge := YearsBetween(date,StrToDate(fDateOfBirth));

if iRealAge>= age then

result := 'APPROVED'

else

result := 'REJECTED';

end;

**// Provided Code - do not change**

functionTPlayer.toString: String;

var

sOut, sApp: String;

begin

sOut := 'Name: ' + fName + #13;

sOut := sOut + 'Email: ' + fEmail + #13;

sOut := sOut + 'Date of Birth: ' + fDateOfBirth + #13;

iffApproved then

sApp := 'YES'

elsesApp := 'NO';

sOut := sOut + 'Approved: ' + sApp + #13;

result := sOut;

end;

end.

**Main Form Unit**

unit question3\_U;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,Dialogs, StdCtrls, player\_U, ComCtrls;

type

TfrmQuestion3 = class(TForm)

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

edtName: TEdit;

edtEmail: TEdit;

edtDateOfBirth: TEdit;

btnSubmitDetails: TButton;

Label6: TLabel;

Label7: TLabel;

btnApprove: TButton;

edtAge: TEdit;

btnAgeTest: TButton;

edtAgeResult: TEdit;

redDisplay: TRichEdit;

btnRegister: TButton;

Label5: TLabel;

edtToday: TEdit;

Label8: TLabel;

procedurebtnSubmitDetailsClick(Sender: TObject);

procedurebtnAgeTestClick(Sender: TObject);

procedurebtnApproveClick(Sender: TObject);

procedureFormShow(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

frmQuestion3: TfrmQuestion3;

objPlayer :TPlayer;

implementation

{$R \*.dfm}

// =====================================================================

**// Question 3.2.1**

// =====================================================================

procedure TfrmQuestion3.btnSubmitDetailsClick(Sender: TObject);

begin

objPlayer :=

TPlayer.create(edtName.text,edtEmail.text,edtDateOfBirth.text);

redDisplay.Clear;

redDisplay.Lines.Add(objPlayer.toString);

edtAgeResult.clear;

end;

// =====================================================================

**// Question 3.2.2**

// =====================================================================

procedure TfrmQuestion3.btnAgeTestClick(Sender: TObject);

begin

edtAgeResult.Text := objPlayer.testAge(StrToInt(edtAge.Text));

end;

// =====================================================================

**// Question 3.2.3**

// =====================================================================

procedure TfrmQuestion3.btnApproveClick(Sender: TObject);

begin

if (edtAgeResult.Text = 'APPROVED') AND

(objPlayer.getEmail <> 'ERROR') then

objPlayer.setApproved(true)

else

begin

showMessage('The player''s age or e-mail has been rejected.');

objPlayer.setApproved(false);

end;

redDisplay.Clear;

redDisplay.Lines.Add(objPlayer.toString);

end;

**// Provided code**

procedure TfrmQuestion3.FormShow(Sender: TObject);

begin

edtToday.text := DateToStr(Date);

end;

end.

**ANNEXURE H: SOLUTION FOR QUESTION 4**

unit Question4\_U;

interface

uses

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

type

TfrmQuestion4 = class(TForm)

Label1: TLabel;

redDisplay: TRichEdit;

Label2: TLabel;

btnload: TButton;

btnRemoveDuplicates: TButton;

btnMasterPlayers: TButton;

procedurebtnloadClick(Sender: TObject);

procedure display;

procedurebtnRemoveDuplicatesClick(Sender: TObject);

procedurebtnMasterPlayersClick(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

frmQuestion4: TfrmQuestion4;

arrNames: array [1..50] of String;

arrScores: array [1..50] of Integer;

iCount: Integer = 0;

arrMasters: array[1..30] of String;

iMastCount: Integer;

implementation

{$R \*.dfm}

// =====================================================================

**// Question 4.1**

// =====================================================================

procedure TfrmQuestion4.btnRemoveDuplicatesClick(Sender: TObject);

var

arrTempNames: array [1..50] of String;

arrTempScores: array [1..50] of Integer;

i, j, newCount : Integer;

found : Boolean;

begin

newCount := 0;

for i := 1 to iCount do

begin

found := false;

for j := 1 to newCount do

begin

if arrNames[i] = arrTempNames[j] then

begin

found := true;

if arrScores[i] > arrTempScores[j] then

arrTempScores[j] := arrScores[i];

end;

end;

if NOT found then

begin

Inc(newCount);

arrTempNames[newCount] := arrNames[i];

arrTempScores[newCount] := arrScores[i];

end;

end;

iCount := newCount;

for i := 1 to iCount do

begin

arrNames[i] := arrTempNames[i];

arrScores[i] := arrTempScores[i];

end;

display;

end;

// =====================================================================

**// Question 4.2**

// =====================================================================

procedure TfrmQuestion4.btnMasterPlayersClick(Sender: TObject);

var

i, j, iTot, iAvg : Integer;

sTemp : String;

begin

iTot := 0;

for i := 1 to iCount do

iTot := iTot + arrScores[i];

iAvg := Round(iTot / iCount);

iMastCount := 0;

for i := 1 to iCount do

ifarrScores[i] >iAvg then

begin

Inc(iMastCount);

arrMasters[iMastCount] := arrNames[i] + '#' +

IntToStr(arrScores[i]);

end;

for i := 1 to iMastCount - 1 do

for j := i + 1 to iMastCount do

if arrMasters[i] > arrMasters[j] then

begin

sTemp := arrMasters[i];

arrMasters[i] := arrMasters[j];

arrMasters[j] := sTemp;

end;

redDisplay.Clear;

redDisplay.Lines.Add('MASTER PLAYERS');

redDisplay.Lines.Add('Scores above an average of ' + IntToStr(iAvg));

for i := 1 to iMastCount do

redDisplay.Lines.Add(arrMasters[i]);

end;

**// Provided Code**

procedure TfrmQuestion4.btnloadClick(Sender: TObject);

var

tScoresFile: TextFile;

sLine : String;

p: Integer;

begin

iCount := 0;

AssignFile(tScoresFile, 'Scores.txt');

Reset(tScoresFile);

while NOT EOF(tScoresFile) do

begin

Inc(iCount);

Readln(tScoresFile, sLine);

p := Pos(',',sLine);

arrNames[iCount] := Copy(sLine,1,p-1);

arrScores[iCount] := StrToInt(Copy(sLine,p+1));

end;

CloseFile(tScoresFile);

display;

end;

procedure TfrmQuestion4.display;

var

i : Integer;

begin

redDisplay.clear;

redDisplay.Paragraph.TabCount := 1;

redDisplay.Paragraph.Tab[0] := 120;

redDisplay.Lines.Add('NAME' + #9 + 'SCORE');

for i := 1 to iCount do

begin

redDisplay.Lines.Add(arrNames[i] + #9 + IntToStr(arrScores[i]));

end;

end;

end.