TRIALS 2023

SUGGESTED ANSWER

QUESTION 1:

```
procedure TfrmQuestion1.btnQuestion1_1Click(Sender: TObject);   8
begin
  //Question 1.1
  imgPic.Stretch := true;✓
  imgPic.Width := 377; ✓
  lblWelcome.Caption := Welcome to Comic Con 2023; ✓
  lblWelcome.Font.Style✓ := [fsUnderline]; ✓
  lblWelcome.Font✓.Name✓ := 'Elephant'; ✓
end;
```

```
procedure TfrmQuestion1.btnQuestion1_2Click(Sender: TObject); 5
const
  GameCost = 59.99;    //Provided
var
  iGame : integer;    //Provided
  rCost : Real;
begin
//Question 1.2
   iGame := sedGames.Value;✓

   rCost := GameCost*✓ (iGame - iGame div 10✓);

   edtCost.Text := FloatToStrF(rCost✓,ffCurrency✓,10,2);
end;
```

```
procedure TfrmQuestion1.btnQuestion1_3Click(Sender: TObject); 10
var
   iHeight, iLen, iWidth : Integer;
   rVolume, rPerc, rRoof : Real;
//Question 1.3
begin
iHouseHeight := 6; //Provided code

   iHeight := StrToInt✓ (InputBox('Height','Enter height of the roof',''));
   iWidth := StrToInt(InputBox('Width','Enter width of the house',''));
   iLen := StrToInt(InputBox('Length','Enter length of the roof',''));  ✓

   rRoof := 1/2*iHeight*iWidth*iLen; ✓
   rVolume := iHouseHeight * iWidth * iLen✓ + rRoof; ✓
   rPerc := rRoof / rVolume * 100; ✓
   redOutput.Lines.Add✓ ('Total Volume = '+FloatToStrF(rVolume,ffFixed,10,1✓));
   redOutput.Lines.Add('Percentage = '+FloatToStrF(rPerc,ffFixed,10,1)+'%');
✓Labels✓vars
end;
```

```
//Question 1.4                                                    [9]
procedure TfrmQuestion1.btnQuestion1_4Click(Sender: TObject);
   sTerm, sAcronym : string;
   k : integer;
begin
   sTerm := edtTerm.Text; ✓
   sAcronym := upcase(sTerm[1]); ✓

   for k := 2 to length(sTerm) ✓ do✓    May start at 1

      if sTerm[k] =' ' then✓
         sAcronym := sAcronym ✓+ upcase✓ (sTerm[k+1] ✓);
```

Alternative using while loop:

```
sTerm := uppercase☑ (edtTerm.Text) ☑;
while pos(' ',sTerm) > 0☑ do☑        OR    while pos(' ',sTerm) <> 0☑ do☑
```

```
  begin
    ipos := pos(' ',sTerm); ☑
    sAcronym := sAcronym☑ + sterm[ipos+ 1] ☑;
    delete(sterm,1,ipos);
  end;

  lblAcronym.Caption := sAcronym; ✓
end;

procedure TfrmQuestion1.btnQuestion1_5Click(Sender: TObject); var
  inum, k, iCode: integer;
begin
  redCollatz.Lines.Clear;
```

Question 1.5 **17**

```
Val(edtCollatz.Text, inum,icode); ✓


  if (icode > 0) ✓ then      OR { if TryStrToInt(edtCollatz.Text,iNum)☑ = false☑ then}
    begin
      edtCollatz.Clear; ✓
      edtCollatz.setfocus; ✓
      edtCollatz.Color✓ := clYellow;
      Exit; ✓
    end;


  k := 1; ✓
  redCollatz.Lines.Add(inttostr(k) + ': ' + inttostr(inum)); ✓
  while inum <> 1✓ do   ✓ conditional loop
  begin
    if inum mod 2 = 0 ✓ then
      inum := inum div 2 ✓
    else ✓
      inum := (inum * 3) ✓ + 1✓;
    inc(k); ✓
   redCollatz.Lines.Add(inttostr(k) + ': ' + inttostr(inum)); ✓
  end;
```

```
{Alternative:
k := 1;☑
  repeat
    redCollatz.Lines.Add(IntToStr(k)+':'+IntToStr(iNum)); ☑
    if Odd(iNum) then☑
      iNum := iNum * 3 ☑+ 1☑
    else☑
      iNum := iNum div 2; ☑
    inc(k); ☑
  until ☑iNum = 1 ; ☑
  redCollatz.Lines.Add(IntToStr(k)+':'+IntToStr(iNum)) ☑;}
end;
end.
```

QUESTION 2: MARKING GRID – SQL AND DATABASE PROGRAMMING

QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
2.1	**SQL statements**		

2.1.1	**Button [2.1.1 – Collectable Action Figures]**
	`SELECT * ✓`
	`FROM tblActionFigure ✓`
	`WHERE Collectable = TRUE ✓`
	`AND ReleaseDate > ✓ #01/31/1999# ✓`
		5	

2.1.2	**Button [2.1.2 – Companies in the USA]**
	`SELECT CompanyName, CompanyLocation ✓`
	`FROM tblCompany ✓`
	`WHERE CompanyLocation LIKE ''%USA%''✓`
	`ORDER BY CompanyName ✓`
		4	

2.1.3	**Button [2.1.3 – All Toys by Company]**
	`SELECT CompanyName, FigureName, ReleaseDate, Price ✓`
	`FROM tblCompany C, tblFigure F ✓`
	`WHERE CompanyName = '+QuotedStr(sSearch) ✓`
	`AND C.CompanyID = F.CompanyID ✓`
		4	

2.1.4	**Button [2.1.4 – Stock Numbers]**
	`SELECT FigureName, NumberInStock ✓`
	`FROM tblFigure ' ✓`
	`WHERE NumberInStock > ✓`
	`SELECT AVG(NumberInStock) ✓ FROM tblFigure ✓`
	`ORDER BY NumberInStock DESC ✓`
		6	

2.1.5	**Button [2.1.5 – Add Company]**
	`INSERT INTO tblCompany ✓`
	`(CompanyID, CompanyName, CompanyLocation,`
	`FoundingDate, Founder, CompanyEmail) ✓`
	`VALUES ✓ (17, "Mattel", "El Segundo, USA",`
	`#12/14/1982#, "Harold Matson","info@mattel.com")✓`
		4	

	Subtotal:	**23**	

QUESTION 2: MARKING GRID (CONT.)

2.2	**DATABASE MANIPULATION using Delphi code**		
2.2.1	**Button [2.2.1 – Cost of metal figures]** Loop through tblFigure ✓ If the material = metal ✓ put DB into Edit mode ✓ add 10% to price ✓ post change ✓	**5**	
2.2.2	**Button [2.2.2 – Total Toys in stock]** Add the heading Move the pointer to the first record of tblCompany ✓ Loop through tblCompany ✓ Initialise the total number of figures to 0 ✓ Extract the company ID ✓and company name ✓ Move the pointer to the first record of tblFigure ✓ Loop through tblFigure ✓ If CompanyID in tblFigure = CompanyID in tblCompany ✓ Add the number of figures to the total ✓ Output the company name ✓ and the total ✓ in neat columns ✓	**12**	
	Subtotal:	**17**	
	TOTAL SECTION B:	**40**	

QUESTION 3: MARKING GRID – OOP

QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
3.1	**clsRegistration**		

3.1.1	**Constructor Create**
	Declaration ✓
	constructor Create(sName: string; iStandSize: integer);
	
	fName := sName;
	fStandSize := iStandSize; ✓
	fCost := 0;
	fBadges := 0; ✓
	fProfile := ''; ✓
		4	

3.1.2	**procedure calcCost;** Declaration ✓ procedure calcCost; Correct ranges ✓✓ Correct values ✓ assigned to fCost ✓ Multiple by 4 ✓	**6**	
3.1.3	**procedure setProfile(sProfile: string);** Declaration ✓ procedure setProfile(sProfile: string); fProfile := sProfile; ✓	**2**	
3.1.4	**function checkWords: boolean;** Declaration ✓ function checkWords: boolean; Initialise counter ✓ Loop 1 to length(fProfile) ✓ Check each character in the string for a space ✓ Increment counter ✓ Increment counter outside the loop ✓ If statement to check number of words. ✓ Assign result to true Otherwise to false ✓	**8**	
3.1.5	**procedure calcBadges;** Ceil ✓ fStandsize/9 ✓ Assign to fBadges ✓	**3**	
3.1.6	**function getBadges: integer;**	**2**	

	Declaration ✓ function getBadges: integer; result := fBadges; ✓		
	Subtotal:	**25**	

3.2	**Driver class**		
3.2.1	**Button [3.2.1 – Register]** sName := edtName.Text; ✓ iStandSize := strtoint(edtStandSize.Text); ✓ objRegistration := TRegistration.Create ✓ (sName, iStandSize); ✓	**4**	
3.2.2	**Button [3.2.2 – Display cost]** objRegistration ✓.calcCost; ✓ redQ3.Lines.Add ✓ (objRegistration.toString); ✓	**4**	
3.2.3	**Button [3.2.3 – Word Limit]** sProfile := edtProfile.Text; ✓ objRegistration.setProfile(sProfile); ✓ if objRegistration.checkWords = true then ✓ pnlQ3_2_3.Caption := 'Profile approved for the directory' else pnlQ3_2_3.Caption := 'Too many words'; ✓	**4**	
3.2.4	**Button [3.2.4 – Badges]** objRegistration.calcBadges; ✓ redQ3.lines.add ('Total number of badges to be issued: ' + inttostr ✓ (objRegistration.getBadges)); ✓	**3**	
	Subtotal:	**15**	
	TOTAL SECTION C:	**40**	

QUESTION 4: - PROBLEM SOLVING PROGRAMMING

LEARNER:			
Question	**DESCRIPTION**	**MAX. MARK**	**LEARNER'S MARKS**
4.1	**Button [Load Results]** Open file for reading (Assign,Reset) ✓ Loop through the file to read each line ✓ Extract the delimited line to Round number, team 1, team 2, Score 1 and Score 2 ✓✓ determine which team gets 3 point or 0 ✓ locate and update the cell to with score ✓✓	7	
4.2.	**Button [Display Results]** Display round 1 to 5 headings ✓ Outer Loop row 1 to 8 ✓ (nested loop) Initialise output line with school name in arrParticipants✓ Inner Loop column 1 to 5 Add arrResult[row, column] item to output string ✓ Display output line to rich edit ✓	5	
4.3.	**Button [Update Result]** Open text file results.txt for append ✓ Extract itemindex from two combo boxes + 1 to refer for row ✓ Extract scores from spin edits ✓ Check if result = '*' for the corresponding cells ✓ Add values from input as the same format as the text file✓ Write built result line of data to text file✓ Else Display error message that result is already captured✓ Close file ✓	8	

4.4.	**Button [Top 3 teams]** Declare arrTotPoints ✓ Loop row 1 to 8 ✓ (nested loop) Initialize sum variable to 0 ✓ Loop column 1 to 5 Add arrResult[row,column] to sum variable ✓ Store sum to arrTotPoints[row] ✓ Sort arrTotPoints in descending order and the corresponding arrParticipants ✓✓✓ Display the first 3 elements of the sorted arrays as show in the sample output ✓✓	10	
	SECTION D TOTAL		**[30]**

SOLUTION FOR QUESTION 2

```
unit Question2_u;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
  Dialogs, dbConnection_U, ADODB, DB, Grids, DBGrids, StdCtrls,
jpeg, ExtCtrls,
  ComCtrls;

type
    TfrmQuestion2 = class(TForm)
    dbgHeroes: TDBGrid;
    btnRestore: TButton;
    btnQ2_1_1: TButton;
    pgcQ2: TPageControl;
    tsQuestion2_1: TTabSheet;
    tsQuestion2_2: TTabSheet;
    imgSuperhero: TImage;
    imgSuperhero2: TImage;
    dbgActionFigure: TDBGrid;
    redQ2: TRichEdit;
    btnQ2_1_2: TButton;
    btnQ2_1_3: TButton;
    cmbCompany: TComboBox;
    btnQ2_1_4: TButton;
    btnQ2_1_5: TButton;
    btnQ2_2_2: TButton;
    btnQ2_2_1: TButton;
    dbgCompany: TDBGrid;
    procedure Connect;
    procedure FormActivate(Sender: TObject);
    procedure btnRestoreClick(Sender: TObject);
    procedure btnQ2_1_1Click(Sender: TObject);
    procedure btnQ2_1_2Click(Sender: TObject);
    procedure btnQ2_1_3Click(Sender: TObject);
    procedure btnQ2_1_4Click(Sender: TObject);
    procedure btnQ2_1_5Click(Sender: TObject);
    procedure btnQ2_2_2Click(Sender: TObject);
    procedure Tabs;
    procedure btnQ2_2_1Click(Sender: TObject);



  private
   { Private declarations }
```

```
  public
    { Public declarations }
  // declare the components
  objHeroes : TConnection;
  dsrCompany : TDataSource;
  qryHeroes : TADOQuery;
  dsrFigure : TDataSource;
  dsrHeroes : TDataSource;

  // tables for Q2.2
  tblCompany : TADOTable;
  tblFigure : TADOTable;

  end;

var
  frmQuestion2: TfrmQuestion2;

implementation

{$R *.dfm}

{ TForm3 }

procedure TfrmQuestion2.btnQ2_1_1Click(Sender: TObject);
var sSQL : String;
begin

  sSQL := 'SELECT * FROM tblFigure WHERE Collectable = True '+
          'AND ReleaseDate > #01/31/1999#';


  qryHeroes.SQL.Clear;
  qryHeroes.SQL.Add(sSQL);
  qryHeroes.Open;
  TFloatField(qryHeroes.FieldByName('Price')).currency := True;
end;

procedure TfrmQuestion2.btnQ2_1_2Click(Sender: TObject);
var sSQL : String;
begin

  sSQL := 'SELECT CompanyName, CompanyLocation FROM tblCompany
'+
          'WHERE CompanyLocation LIKE ''%USA%'' ORDER BY
CompanyName';

  qryHeroes.SQL.Clear;
```

```
  qryHeroes.SQL.Add(sSQL);
  qryHeroes.Open;
end;

procedure TfrmQuestion2.btnQ2_1_3Click(Sender: TObject);
var
   sSearch, sSQL : String;
begin

  // given code
  sSearch := cmbCompany.Text;

  sSQL := 'SELECT CompanyName, FigureName, ReleaseDate, Price '+
          'FROM tblCompany C, tblFigure F '+
          'WHERE CompanyName = '+QuotedStr(sSearch)+
          'AND C.CompanyID = F.CompanyID ';

  // given code
  qryHeroes.SQL.Clear;
  qryHeroes.SQL.Add(sSQL);
  qryHeroes.Open;
  TFloatField(qryHeroes.FieldByName('Price')).currency := True;
end;

procedure TfrmQuestion2.btnQ2_1_4Click(Sender: TObject);
var sSQL : String;
begin

  sSQL := 'SELECT FigureName, NumberInStock '+
          'FROM tblFigure '+
          'WHERE NumberInStock > '+
          '(SELECT AVG(NumberInStock) FROM tblFigure)'+
          'ORDER BY NumberInStock DESC';

  // given code
  qryHeroes.SQL.Clear;
  qryHeroes.SQL.Add(sSQL);
  qryHeroes.Open;
end;

procedure TfrmQuestion2.btnQ2_1_5Click(Sender: TObject);
var sSQL : String;
begin

  sSQL := 'INSERT INTO tblCompany(CompanyID, CompanyName,
CompanyLocation, '+
          'FoundingDate, Founder, CompanyEmail) '+
          'VALUES(17, "Mattel", "El Segundo, USA", '+
          '#12/14/1982#, "Harold Matson", "info@mattel.com")';
```

```
  // given code
  qryHeroes.SQL.Clear;
  qryHeroes.SQL.Add(sSQL);
  qryHeroes.ExecSQL;

  qryHeroes.SQL.Clear;
  qryHeroes.SQL.Add('SELECT * FROM tblCompany');
  qryHeroes.Open;
  qryHeroes.Last;
end;

procedure TfrmQuestion2.btnQ2_2_1Click(Sender: TObject);
begin

   tblFigure.First;

   while not(tblFigure.Eof) do
   begin
     if tblFigure['Material'] = 'metal' then
     begin
        tblFigure.Edit;
        tblFigure['Price'] := tblFigure['Price'] * 1.1;
        tblFigure.Post;
     end;

     tblFigure.Next;
   end;

   // given code
   tblFigure.First;
   TFloatField(tblFigure.FieldByName('Price')).currency := True;
end;


procedure TfrmQuestion2.btnQ2_2_2Click(Sender: TObject);
var
    iNumFigures : Integer;
    sCompanyID, sCompanyName : String;
begin

    // given code
    redQ2.Lines.Add('Company Name'+#9+'Number in Stock'+#13);


    tblCompany.First;

    while not(tblCompany.Eof) do
    begin
```

```
      iNumFigures := 0;

      sCompanyID := tblCompany['CompanyID'];
      sCompanyName := tblCompany['CompanyName'];
      tblFigure.First;

      while not(tblFigure.Eof) do
      begin

         if tblFigure['CompanyID'] = sCompanyID then
         begin
            iNumFigures := iNumFigures +
tblFigure['NumberInStock'];
         end;
         tblFigure.Next;

      end;

      redQ2.Lines.Add(sCompanyName+#9+IntToStr(iNumFigures));
      tblCompany.Next;
    end;
end;
```

```
{$REGION}

//---------------------------------------------------------------
----------------
// PROVIDED CONNECTION CODE - DO NOT CHANGE!
//---------------------------------------------------------------
----------------

procedure TfrmQuestion2.btnRestoreClick(Sender: TObject);
var
  failFlag: Boolean;
begin

  objHeroes.Disconnect;
  qryHeroes.Close;
  tblCompany.Close;
```

```
  tblFigure.Close;

  DeleteFile('Heroes.mdb');
  CopyFile('Heroesbackup.mdb', 'Heroes.mdb', failFlag);

  objHeroes.DBConnect;
  tblCompany.Open;
  tblFigure.Open;
  qryHeroes.Open;

  ShowMessage('Database restored!');

end;

procedure TfrmQuestion2.Connect;
begin

   qryHeroes := TADOQuery.Create(frmQuestion2);
   qryHeroes.Connection := objHeroes.dbConnection;
   qryHeroes.SQL.Clear;
   qryHeroes.SQL.Add('SELECT * FROM tblCompany');
   qryHeroes.Open;

   tblCompany := TADOTable.Create(frmQuestion2);
   tblCompany.Connection := objHeroes.dbConnection;
   tblCompany.TableName := 'tblCompany';
   tblCompany.Active := True;

   tblFigure := TADOTable.Create(frmQuestion2);
   tblFigure.Connection := objHeroes.dbConnection;
   tblFigure.TableName := 'tblFigure';
   tblFigure.Active := True;

   dsrHeroes := TDataSource.Create(frmQuestion2);
   dsrHeroes.DataSet := qryHeroes;

   dsrCompany := TDataSource.Create(frmQuestion2);
   dsrCompany.DataSet := tblCompany;

   dsrFigure := TDataSource.Create(frmQuestion2);
   dsrFigure.DataSet := tblFigure;

   dbgHeroes.DataSource := dsrHeroes;
   dbgCompany.DataSource := dsrCompany;
   dbgActionFigure.DataSource := dsrFigure;

   Tabs;

   TFloatField(tblFigure.FieldByName('Price')).currency := True;
```

```
end;

procedure TfrmQuestion2.FormActivate(Sender: TObject);
var
  I: Integer;
begin

   objHeroes := TConnection.Create;
   objHeroes.DBConnect;

   Connect;

   cmbCompany.Clear;

   for I := 1 to qryHeroes.RecordCount do
   begin
     cmbCompany.Items.Add(qryHeroes['CompanyName']);

     qryHeroes.Next;
   end;

   cmbCompany.Sorted := True;
   cmbCompany.ItemIndex := 0;
   qryHeroes.SQL.Clear;
end;

procedure TfrmQuestion2.Tabs;
begin
  redQ2.Paragraph.TabCount := 1;
  redQ2.Paragraph.Tab[0] := 100;
end;

{$ENDREGION}
end.
```

SOLUTION FOR QUESTION 3

```
// Enter your name and surname
unit clsRegistration;

interface

uses sysutils, math;

type
  TRegistration = class(TObject)
  private
    { private declarations }
    // Provided Code
    fName: string;
    fStandSize: integer;
    fCost: real;
    fProfile: string;
    fBadges: integer;

  public
    { public declarations }

    // Provided code
    function toString: string;

    // Question 3.1.1, 3.1.2, 3.1.3, 3.1.4, 3.1.5, 3.1.6
    constructor Create(sName: string; iStandSize: integer);
    procedure calcCost;
    procedure setProfile(sProfile: string);
    function checkWords: boolean;
    procedure calcBadges;
    function getBadges: integer;
  end;

implementation

{ TRegistration }

// Question 3.1.1
constructor TRegistration.Create(sName: string; iStandSize:
integer);
begin
  fName := sName;
  fStandSize := iStandSize;
  fCost := 0;
  fBadges := 0;
  fProfile := '';
end;
```

```
// Question 3.1.2
procedure TRegistration.calcCost;
begin
  case fStandSize of
    1 .. 18:
      fCost := 3690 * 4;
    19 .. 36:
      fCost := 3585 * 4;
  else
    fCost := 3447 * 4;
  end;
end;

// Question 3.1.3
procedure TRegistration.setProfile(sProfile: string);
begin
  fProfile := sProfile;
end;

// Question 3.1.4
function TRegistration.checkWords: boolean;
var
  iWords, i: integer;
begin
  iWords := 0;
  for i := 1 to length(fProfile) do
  begin
    if fProfile[i] = ' ' then
      inc(iWords);
  end;
  inc(iWords);
  if iWords <= 50 then
    result := true
  else
    result := false;
end;

// Question 3.1.5
procedure TRegistration.calcBadges;
begin
  fBadges := ceil(fStandSize / 9);
end;

// Question 3.1.6
function TRegistration.getBadges: integer;
begin
  result := fBadges;
end;
```

```
// Given code
function TRegistration.toString: string;
begin
  result := fName + #13 + 'Stand size: ' + inttostr(fStandSize)
+ ' sqm'
    + #13 + 'Total cost for 4 days: ' + floattostrf(fCost,
ffcurrency, 10, 2)+#13;
end;

end.

//Question 3
//Enter your name and surname
unit Question3_u;

interface

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

type
  TfrmQuestion3 = class(TForm)
    redQ3: TRichEdit;
    Panel1: TPanel;
    lblExhibitor: TLabel;
    edtName: TEdit;
    lblSqm: TLabel;
    edtStandSize: TEdit;
    btnq3_2_1: TButton;
    btnq3_2_2: TButton;
    Panel2: TPanel;
    lblProfile: TLabel;
    edtProfile: TEdit;
    btnq3_2_3: TButton;
    pnlQ3_2_3: TPanel;
    btnq3_2_4: TButton;
    procedure btnq3_2_1Click(Sender: TObject);
    procedure btnq3_2_2Click(Sender: TObject);
    procedure btnq3_2_3Click(Sender: TObject);
    procedure btnq3_2_4Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }

    //Provided Code
```

```
    objRegistration : TRegistration;
  end;

var
  frmQuestion3: TfrmQuestion3;

implementation

{$R *.dfm}

procedure TfrmQuestion3.btnq3_2_1Click(Sender: TObject);
var sName : string;
    iStandSize : integer;
begin
//Question 3.2.1
   sName := edtName.Text;
   iStandSize := strtoint(edtStandSize.Text);

   objRegistration := TRegistration.Create(sName, iStandSize);

   //Provided code
   showmessage('Object has been instantiated');
end;

procedure TfrmQuestion3.btnq3_2_2Click(Sender: TObject);
begin
  //Question 3.2.2
  objRegistration.calcCost;
  redQ3.Lines.Add(objRegistration.toString);
end;

procedure TfrmQuestion3.btnq3_2_3Click(Sender: TObject);
var sProfile : string;
begin
   //Question 3.2.3
   sProfile := edtProfile.Text;
   objRegistration.setProfile(sProfile);
   if objRegistration.checkWords = true then
   pnlQ3_2_3.Caption := 'Profile approved for the directory'
   else pnlQ3_2_3.Caption := 'Too many words';
end;

procedure TfrmQuestion3.btnq3_2_4Click(Sender: TObject);
begin
   //Question 3.2.4
   objRegistration.calcBadges;
   redQ3.lines.add('Total number of badges to be issued: ' +
inttostr(objRegistration.getBadges));
end;
```

SOLUTION FOR QUESTION 4

```
//MEMO
unit Q4;

interface

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

type
  TQuest4_U = class(TForm)
    redDisplay: TRichEdit;
    btnTop3: TButton;
    btnLoadResult: TButton;
    Panel1: TPanel;
    Image1: TImage;
    GroupBox1: TGroupBox;
    btnUpdateResults: TButton;
    cmbTeam1: TComboBox;
    cmbTeam2: TComboBox;
    Label1: TLabel;
    sedScore1: TSpinEdit;
    sedScore2: TSpinEdit;
    Label2: TLabel;
    Label3: TLabel;
    btnDisplay: TButton;
    sedRound: TSpinEdit;
    Label4: TLabel;
    redQ4_4: TRichEdit;
    procedure FormActivate(Sender: TObject);
    procedure btnLoadResultClick(Sender: TObject);
    procedure btnUpdateResultsClick(Sender: TObject);
    procedure btnDisplayClick(Sender: TObject);
    procedure btnTop3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

const
//Provided code
  NumPart = 8;
```

```
var
  Quest4_U: TQuest4_U;

  //Provided code
  arrParticipants: array [1 .. 8] of string = (
    'NWD',
    'OIS',
    'DHS',
    'GCB',
    'DGH',
    'NGH',
    'WVB',
    'EDC'
  );
  arrResult: array [1 .. 8, 1 .. 5] of String;


  {arrResult2: array [1 .. 8, 1 .. 5] of String = (('3', '3',
'3', '3', '0'),
    ('0', '3', '0', '*', '3'), ('3', '0', '0', '*', '3'),
    ('3', '0', '*', '0', '3'), ('3', '3', '0', '3', '0'),
    ('0', '3', '3', '0', '0'), ('0', '0', '*', '3', '3'),
    ('0', '0', '3', '0', '0'));
      }
implementation

{$R *.dfm}

procedure TQuest4_U.btnDisplayClick(Sender: TObject);
var
  r: Integer;
  c: Integer;
  sLine: String;
begin
  //Provided code
  redDisplay.Clear;

  //Question 4.2
  redDisplay.Text := 'School ' + #9 + 'R1' + #9 + 'R2' + #9 +
'R3' + #9 +
    'R4' + #9 + 'R5';
  for r := 1 to 8 do
  begin
    sLine := arrParticipants[r];
    for c := 1 to 5 do
    begin
      sLine := sLine + #9 + arrResult[r, c];
    end;
    redDisplay.Lines.Add(sLine);
```

```
  end;

end;

procedure TQuest4_U.btnLoadResultClick(Sender: TObject);
var
  inf: TextFile;
  sLine: String;
  sTeam1, sTeam2, sR1, sR2: String;
  I, iRound: Integer;
  iPos: Integer;
begin
  //Question 4.1
  AssignFile(inf, 'Results.txt');
  reset(inf);
  while not eof(inf) do
  begin
    readln(inf, sLine);
    iRound := StrToInt(copy(sLine, 1, pos('#', sLine) - 1));
    delete(sLine, 1, pos('#', sLine));
    sTeam1 := copy(sLine, 1, pos('#', sLine) - 1);
    delete(sLine, 1, pos('#', sLine));
    sTeam2 := copy(sLine, 1, pos('#', sLine) - 1);
    delete(sLine, 1, pos('#', sLine));
    sR1 := copy(sLine, 1, pos('#', sLine) - 1);
    delete(sLine, 1, pos('#', sLine));
    sR2 := sLine;

    for I := 1 to 8 do
    begin
      if sTeam1 = arrParticipants[I] then
      begin
        if StrToInt(sR1) > StrToInt(sR2) then
          arrResult[I, iRound] := '3'
        else
          arrResult[I, iRound] := '0'
      end;
    end;
    for I := 1 to 8 do
    begin
      if sTeam2 = arrParticipants[I] then
      begin
        if StrToInt(sR2) > StrToInt(sR1) then
          arrResult[I, iRound] := '3'
        else
          arrResult[I, iRound] := '0'
      end;
    end;
  end;
```

```
  closeFile(inf);
  showmessage('Results loaded to a arrResults successfully.');
end;

procedure TQuest4_U.btnTop3Click(Sender: TObject);
var
  arrTot: array [1 .. 8] of Integer;
  r: Integer;
  c: Integer;
  iSum: Integer;
  I: Integer;
  j: Integer;
  iTemp: Integer;
  sTemp: String;
begin
  //Question 4.4

  for r := 1 to 8 do
  begin
    iSum := 0;
    for c := 1 to 5 do
    begin
      iSum := iSum + StrToInt(arrResult[r, c]);
    end;
    arrTot[r] := iSum;
  end;
  for I := 1 to 7 do
    for j := I + 1 to 8 do
    begin
      if arrTot[I] < arrTot[j] then
      begin
        iTemp := arrTot[I];
        arrTot[I] := arrTot[j];
        arrTot[j] := iTemp;

        sTemp := arrParticipants[I];
        arrParticipants[I] := arrParticipants[j];
        arrParticipants[j] := sTemp;
      end;
    end;
  for I := 1 to 3 do
  begin
    redQ4_4.Lines.Add('Position ' + inttostr(I) + ': ' +
arrParticipants[I]
        + ' with ' + inttostr(arrTot[I]));
  end;
end;

procedure TQuest4_U.btnUpdateResultsClick(Sender: TObject);
```

```
var
  outF: TextFile;
begin
  //Question 4.3

  AssignFile(outF, 'Results.txt');
  Append(outF);
  if (arrResult[cmbTeam1.ItemIndex + 1, sedRound.Value] = '*')
and
    (arrResult[cmbTeam2.ItemIndex + 1, sedRound.Value] = '*')
then
  begin
    writeln(outF,
      sedRound.Text + '#' + cmbTeam1.Text + '#' + cmbTeam2.Text
+ '#' +
        sedScore1.Text + '#' + sedScore2.Text);
    showmessage('Result captured successfully in Result.txt
file.');
  end
  else
    showmessage('Error! Incorrect fixture.Try again.');
  closeFile(outF);

end;

// **************SUPPLIED CODE*******************

procedure TQuest4_U.FormActivate(Sender: TObject);
var
  I, r, c: Integer;
begin
  // format display
  redDisplay.Paragraph.TabCount := 4;
  redDisplay.Paragraph.Tab[0] := 100;
  redDisplay.Paragraph.Tab[1] := 150;
  redDisplay.Paragraph.Tab[2] := 200;
  redDisplay.Paragraph.Tab[3] := 250;
  redDisplay.Paragraph.Tab[4] := 300;

  // Clear any existing items
  cmbTeam1.Items.Clear;
  cmbTeam2.Items.Clear;

  // Add participanting schools array elements to the ComboBox
  for I := Low(arrParticipants) to High(arrParticipants) do
  begin
    cmbTeam1.Items.Add(arrParticipants[I]);
    cmbTeam2.Items.Add(arrParticipants[I]);
  end;
```

```
  // initialize all results to * (not captured yet)
  for r := 1 to 8 do
    for c := 1 to 5 do
      arrResult[r, c] := '*';

end;
end.
```