



basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

NOVEMBER 2013

MEMORANDUM

MARKS: 120

This memorandum consists of 34 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 of whom are required to attend a rigorous standardisation meeting to ensure that the guidelines are consistently interpreted and applied in the marking of candidates' scripts.
- It is acknowledged that there may be different views about some matters of emphasis or detail in the guidelines, and different interpretations of the application thereof.
- Note that learners who provide an alternate correct solution to that given in the marking guidelines will have been given full credit for the relevant question.
- **Annexures A, B and C** (pages 3-6) include the marking grid for each question for using either one of the two programming languages.
- **Annexures D, E, F and G** (pages 7-19) contain the solutions for Delphi for QUESTIONS 1 to 3 in programming code.
- **Annexures H, I, J and K** (pages 20-32) contain the solutions for Java for QUESTIONS 1 to 3 in programming code.
- Copies of Annexures A, B and C should be made for each learner and completed during the marking session

ANNEXURE A

QUESTION 1: MARKING GRID – PROGRAMMING AND DATABASE

GENERAL NOTES:

- Only penalise for the incorrect use of quotes ONCE. Repeated incorrect use of quotes in follow up questions doesn't get penalised.
- The use of = for strings, the use of LIKE may be used as alternative.

CENTRE NUMBER:		EXAMINATION NUMBER:	
QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
1.1	Query: Correct list of fields (or *)✓; correct table✓; ORDER BY correct fields in correct order✓	3	
	SQL: SELECT * FROM tblResults ORDER BY TypeOfDance, RoutineNo Desc		
1.2	Query: Correct fields & table✓; WHERE Correct Score✓ both weeks✓ correct operator used (OR/IN) for the weeks✓	4	
	SQL: SELECT RoutineNo, Week, TypeOfDance, Score FROM tblResults WHERE (Score BETWEEN 25 AND 35) AND (Week=5 OR Week=9) Alternative: (Score >= 25) AND (Score <= 35) (Score > 24) AND (Score < 36) Score IN [25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35] Week IN [5, 9] Score >= 25 And Score <= 35 And (Week = 5 Or Week = 9) (Score >= 25 And (Week = 5 Or Week = 9)) And (Score <= 35 And (Week = 5 Or Week = 9)) Score >= 25 And Score <= 35 And Week IN (5,9) Score IN (25,26,27,28,29,30,31,32,33,34,35) Or Week IN (5,9) Score BETWEEN 25 And 35 and Week = 5 OR Score BETWEEN 25 And 35 and Week = 9 NOTE: Check the correctness of alternative use of intervals		

QUESTION 1: MARKING GRID – PROGRAMMING AND DATABASE – continue

1.3	<p>Query: Correct field & table✓; Count✓; AS NumberOfPerformances✓ WHERE TypeOfDance equals user input✓ GROUP BY TypeOfDance✓</p> <p>SQL: (D) SELECT TypeOfDance, Count(*) AS NumberOfPerformances FROM tblResults WHERE TypeOfDance = '"+ sX + "' GROUP BY TypeOfDance</p> <p>SQL: (J) SELECT TypeOfDance, Count(*) AS NumberOfPerformances FROM tblResults WHERE TypeOfDance = '"+ sX + "' GROUP BY TypeOfDance</p> <p>Alternative: May use Count(<field name>)</p> <p>NOTE: The use of Distinct is not allowed</p>	5	
1.4	<p>Query: Correct fields✓; both tables✓; WHERE linking tables on DanceCoupleID✓; professional dancers with AND operator✓; LIKE Love% ✓; OR-operators with correct use of brackets✓; LIKE %you%✓</p> <p>SQL: (D) SELECT Song, DancePartner1, DancePartner2 FROM tblDanceCouples, tblResults WHERE tblResults.DanceCoupleID = tblDanceCouples.DanceCoupleID AND (ProfessionalDancers = "B") AND ((Song Like "Love%") OR (Song LIKE "%you%"))</p> <p>SQL: (J) SELECT Song, DancePartner1, DancePartner2 FROM tblDanceCouples, tblResults WHERE tblResults.DanceCoupleID = tblDanceCouples.DanceCoupleID AND (ProfessionalDancers = 'B') AND ((Song Like 'Love%') OR (Song LIKE '%you%'))</p> <p>Alternative: Make use of ALIASES for table names Make use of INNER JOIN statement</p> <p>(D) The use of LEFT(Song, 4) = "Love" (J) The use of LEFT(Song, 4) = 'Love'</p> <p>NOTE: The use of * instead of % subtract only ONE mark</p>	7	

QUESTION 1: MARKING GRID – PROGRAMMING AND DATABASE – continue

1.5	<p>Query: Correct field & correct table✓; Format to THREE decimals✓; sum(Score)✓; divide by Count(*)✓; AS AverageScore✓; GROUP BY DanceCoupleID✓</p> <p>SQL: (D) SELECT DanceCoupleID, Format((Sum(Score)/Count(*)), "0.000") AS AverageScore FROM tblResults Group BY DanceCoupleID</p> <p>SQL: (J) SELECT DanceCoupleID, Format((Sum(Score)/Count(*)), '0.000') AS AverageScore FROM tblResults Group BY DanceCoupleID</p> <p>Alternative: Use of different formatting strings, e.g. "#.000"</p> <p>The use of AVG(Score) – TWO marks</p> <p>Round(<calculation>, 3) instead of Format</p>	6	
1.6	<p>Query: Correct fields✓ from both tables✓; WHERE linking tables on DanceCoupleID✓; Result equals Eliminated✓; No duplicates – check for use of week 12 (included must have DISTINCT/Group by) (less than 12 – no distinct/group by required)✓</p> <p>SQL: (D) SELECT DISTINCT DancePartner1, DancePartner2 FROM tblResults, tblDanceCouples WHERE (tblResults.DanceCoupleID = tblCouples.DanceCoupleID) AND (Result LIKE "Eliminated") AND (Week < 12)</p> <p>SQL: (J) SELECT DISTINCT DancePartner1, DancePartner2 FROM tblResults, tblDanceCouples WHERE (tblResults.DanceCoupleID = tblCouples.DanceCoupleID) AND (Result LIKE 'Eliminated') AND (Week < 12)</p> <p>Alternative: Make use of ALIASES for table names Alternative: make use of INNER JOIN statement</p> <p>NOTE: If week 12 is included then a DISTINCT/GROUP BY must be used.</p>	5	
1.7	<p>Query: UPDATE table✓; SET Result to WINNERS✓; WHERE Second Round✓; AND ✓; CoupleID is 8✓</p> <p>SQL: (D) UPDATE tblResults SET Result="WINNERS" WHERE Round = 2 AND DanceCoupleID = 8</p> <p>SQL: (J) UPDATE tblResults SET Result='WINNERS' WHERE Round = 2 AND DanceCoupleID = 8</p> <p>NOTE: The use of Week is optional</p>	5	
	TOTAL:	35	

ANNEXURE B

QUESTION 2: MARKING GRID – OBJECT-ORIENTED PROGRAMMING

GENERAL NOTES:

- If the learner changed any given data type (e.g. character to string) penalise with ONE mark.
- Syntax error (e.g. ;) penalise only ONCE.
- In Java the use of single = in stead of == penalise only ONCE.
- **NO USE OF OBJECT CLASS:** Q2.1.1: 0 marks; Q2.1.2: maximum of 6 marks; Q2.1.3: Maximum of 5 marks; Q2.1.4: maximum of 3 marks

CENTRE NUMBER:		EXAMINATION NUMBER:	
QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
2.1.1	Parameterised CONSTRUCTOR All three correct parameters ✓ with correct data types ✓ Set dance partner name 1 ✓ set dance partner name 2 ✓ Set professional status ✓	5	
2.1.2	getWeighting METHOD: Correct method return type (integer) ✓; Structure of if/case/switch ✓ Testing for A and assign weighting value of 1 ✓; Testing for B and C and assign weighting value of 2 ✓ ✓ {using OR/multiple If's}; Testing for D and assign weighting value of 3 ✓; Assign a weighting value of 0 for an invalid dance status characters/default value of zero ✓; Return the content of the assigned variable ✓ NOTE: Accept return type string ONLY if conversion is used in call statements.	8	
2.1.3	calculateFinalScore METHOD: integer array parameter ✓; loop through array OR refer to individual elements in array ✓; call getWeighting method OR used weighting as an attribute of the class ✓; multiply array element 1 st ✓ and 3 rd by weighting ✓; add the scores ✓; return total score ✓	7	
2.1.4	toString METHOD: Correct method header and return type (string) ✓; Construct a line with correct label ✓; names of both partners with &-character ✓; Construct a line with professional dance status (call method/attribute) ✓ on next line ✓; Return string ✓	6	

QUESTION 2: MARKING GRID – OBJECT-ORIENTED PROGRAMMING – continue

2.2.1	MENU OPTION A: Prompt user for input✓; Input two strings for partner values✓; Input and use the first character of professional status✓; Declare object local/global in unit/class✓ and instantiate DanceCouple object✓; with values entered as arguments✓; in the correct order✓; Display the information of the DanceCouple object✓	8	
2.2.2	MENU OPTION B: Declare an array of integer to keep four scores✓; Get FOUR inputs from user✓; Validate FOUR input values in range 1 to 10✓; Assign a zero if value outside range✓; <i>1st line of output:</i> Display label✓ and call get methods for the names of partners/use global variables for names✓; <i>2nd line of output:</i> Display label and four values ✓ <i>3rd line of output:</i> Display label and call getWeighting method to display the weighting value ✓ <i>4th line of output:</i> Display label and call calcFinalScore method✓ with correct parameter ✓	10	
2.2.3	MENU OPTION C: Text file: ✓ {Delphi: AssignFile & Rewrite Java: Create object to use as a File writer} Construct line of text containing the names of both dance partners✓ and their final score✓ in comma separated format ✓; Write line of text to text file✓; CloseFile/Close writer object✓; Confirmation message to the user ✓ NOTE (Delphi): Accept the use of SaveToFile method of the richedit.	7	
	TOTAL:	51	

ANNEXURE C

QUESTION 3: MARKING GRID - PROBLEM-SOLVING PROGRAMMING

GENERAL NOTES:

- Syntax error (e.g. ;) penalise only ONCE.

CENTRE NUMBER:		EXAMINATION NUMBER:	
QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
3.1	MENU OPTION A: Using appropriate data structure (1D/2D/object array) for all fourteen couples✓; Using a Loop until all user input valid✓: User input to indicate four couples eliminated✓; Validate each value to be in the correct range✓ Use VAL/try...except/catch to validate input not characters✓; Text file: {DELPHI: AssignFile, Reset JAVA: Create object to read from file} ✓; Loop through the lines of text in the file✓ Read line from text file✓ Split data (name✓, age✓, contact number✓, couple number✓) If valid age✓ AND dance couple not eliminated ✓ <i>valid vote:</i> Increment valid counter✓ If international vote✓ based on not +27✓ add label ✓ <i>NOT valid vote:</i> Increment invalid counter ✓ Display output concatenated line of info within the loop✓ Display total valid votes✓; Display total invalid votes✓;	22	
3.2	MENU OPTION B: Using appropriate data structure(s) for counting votes✓ Loop✓ to count votes for each couple ✓ (<i>can be done in question 3.1 inside the IF-statement for a valid vote</i>) Display headings and labels before loop✓ (<i>also accept as part of question 3.1</i>) Loop✓ to display votes received by each couple✓ or the label✓ "(Eliminated)" when couple is eliminated Initialise lowest value✓; determine lowest total votes✓ ignoring couples already eliminated✓ inside a loop OR sorting an array Display all the dance couple(s) who may be eliminated next✓ with label✓	12	
TOTAL:		34	

SUMMARY OF LEARNER'S MARKS:

	QUESTION 1	QUESTION 2	QUESTION 3	GRAND TOTAL
MAX. MARKS	35	51	34	120
LEARNER'S MARKS				

ANNEXURE D: SOLUTION – QUESTION 1: DELPHI

```
unit Question1U_MEMO;
{A solution for Question 1}
interface

uses
    Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
    Dialogs, StdCtrls, DB, ADODB, Grids, DBGrids, ExtCtrls, Buttons, Menus;

type
    TfrmQ1 = class(TForm)
        qryRec: TADOQuery;
        dsrQry: TDataSource;
        grdQ1: TDBGrid;
        mnuMain: TMainMenu;
        mnuOptionA: TMenuItem;
        mnuOptionB: TMenuItem;
        mnuOptionC: TMenuItem;
        mnuOptionD: TMenuItem;
        mnuOptionE: TMenuItem;
        mnuOptionF: TMenuItem;
        mnuOptionG: TMenuItem;
        mnuQuit: TMenuItem;
        procedure mnuOptionAClick(Sender: TObject);
        procedure mnuOptionBClick(Sender: TObject);
        procedure mnuOptionCClick(Sender: TObject);
        procedure mnuOptionDClick(Sender: TObject);
        procedure mnuOptionEClick(Sender: TObject);
        procedure mnuOptionFClick(Sender: TObject);
        procedure mnuOptionGClick(Sender: TObject);
        procedure mnuQuitClick(Sender: TObject);
    private
        { Private declarations }
    public
        { Public declarations }
    end;

var
    frmQ1: TfrmQ1;

implementation

{$R *.dfm}
//=====
procedure TfrmQ1.mnuOptionAClick(Sender: TObject);
begin
    qryRec.Close;
    qryRec.SQL.Text := 'SELECT * FROM tblResults ' +
        'ORDER BY TypeOfDance, RoutineNo Desc';
    qryRec.Open;
end;
//=====
procedure TfrmQ1.mnuOptionBClick(Sender: TObject);
begin
    qryRec.Close;
    qryRec.SQL.Text := 'SELECT RoutineNo, Week, TypeOfDance, Score ' +
        'FROM tblResults WHERE (Score BETWEEN 25 AND 35) AND ' +
        '(Week=5 OR Week=9)';
    qryRec.Open;
end;
//=====
Copyright reserved
```

```

procedure TfrmQ1.mnuOptionCClick(Sender: TObject);
var
  sX : String;
begin
  sX := INPUTBOX('Question 1', 'Enter the TYPE of the dance e.g. Rumba.', '');
  qryRec.Close;
  qryRec.SQL.Text := 'SELECT TypeOfDance, Count(*) AS NumberOfPerformances ' +
    'FROM tblResults WHERE TypeOfDance = "' + sX +
    '" GROUP BY TypeOfDance';

  qryRec.Open;
end;
//=====
procedure TfrmQ1.mnuOptionDClick(Sender: TObject);
begin
  qryRec.Close;
  qryRec.SQL.Text := 'SELECT Song, DancePartner1, DancePartner2 ' +
    'FROM tblCouples, tblResults ' +
    'WHERE tblResults.DanceCoupleID=tblCouples.DanceCoupleID AND ' +
    '(ProfessionalDancers = "B") AND ' +
    '((Song Like "Love%") OR (Song LIKE "%you%"))';

  qryRec.Open;
end;
//=====
procedure TfrmQ1.mnuOptionEClick(Sender: TObject);
begin
  qryRec.Close;
  qryRec.SQL.Text := 'SELECT DanceCoupleID, ' +
    'Format((Sum(Score)/Count(*)), "0.000") AS ' +
    'AverageScore FROM tblResults Group BY DanceCoupleID';

  qryRec.Open;
end;
//=====
procedure TfrmQ1.mnuOptionFClick(Sender: TObject);
begin
  qryRec.Close;
  qryRec.SQL.Text := 'SELECT DancePartner1, DancePartner2 ' +
    'FROM tblResults, tblCouples ' +
    'WHERE (tblResults.DanceCoupleID=tblCouples.DanceCoupleID) ' +
    'AND (Result LIKE "Eliminated") AND (Week < 12) ' +
    'GROUP BY DancePartner1, DancePartner2';

  qryRec.Open;
end;
//=====
procedure TfrmQ1.mnuOptionGClick(Sender: TObject);
begin
  qryRec.Close;
  qryRec.SQL.Text := 'UPDATE tblResults SET Result = "WINNERS" ' +
    'WHERE Round = 2 AND DanceCoupleID = 8';

  qryRec.ExecSQL;
  MessageDlg('Records Processed Successfully', mtInformation, [mbOK], 0);
end;
//=====
procedure TfrmQ1.mnuQuitClick(Sender: TObject);
begin
  Application.Terminate;
end;

end.

```

ANNEXURE E: SOLUTION – QUESTION 2: DELPHI

2.1. GIVEN DANCE COUPLE CLASS UNIT:

```
unit uDanceCouple_MEMO;
{A solution for question 2 - class unit.}
interface

type

    TScoresArray = array[1..4] of integer;

    TDanceCouple = class(TObject)
    private
        fDancePartner1,
        fDancePartner2 : String;
        fProfessional : char;
    public
        constructor Create(); overload;
        constructor Create(sPartner1, sPartner2 : String; cProf : Char);
overload;
        function GetDancePartner1 : String;
        function GetDancePartner2 : String;
        function GetProfessional: char;
        function GetWeighting : integer;
        function CalcFinalScore(arrJudgeScores : TScoresArray) : integer;
        function toString : String;
    end;

implementation
{ TDanceCouple }

constructor TDanceCouple.Create();
begin

end;

constructor TDanceCouple.Create(sPartner1, sPartner2: String;
    cProf: char);
begin
    fDancePartner1 := sPartner1;
    fDancePartner2 := sPartner2;
    fProfessional := cProf;
end;

function TDanceCouple.GetDancePartner1: String;
begin
    Result := fDancePartner1;
end;

function TDanceCouple.GetDancePartner2: String;
begin
    Result := fDancePartner2;
end;

function TDanceCouple.GetProfessional : char;
begin
    Result := fProfessional;
end;

function TDanceCouple.GetWeighting: integer;
```

```
begin
  case fProfessional of
    'A' : Result := 1;
    'B', 'C' : Result := 2;
    'D' : Result := 3;
  else
    Result := 0;
  end; //case
end;

function TDanceCouple.CalcFinalScore(arrJudgeScores: TScoresArray): integer;
begin
  Result := (GetWeighting * arrJudgeScores[1]) + arrJudgeScores[2] +
    (GetWeighting * arrJudgeScores[3]) + arrJudgeScores[4];
end;

function TDanceCouple.toString: String;
begin
  Result := 'Couple: ' + GetDancePartner1 + ' & ' + GetDancePartner2 + #13 +
    'Professional dance status: ' + GetProfessional;
end;
end.
```

2.2 FORM UNIT – QUESTION 2:

```
unit Question2U_MEMO;
{A possible solution for question 2}

interface

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

type
  TfrmQ2 = class(TForm)
    mnuMain: TMainMenu;
    mnuOptionA: TMenuItem;
    mnuOptionB: TMenuItem;
    mnuOptionC: TMenuItem;
    mnuQuit: TMenuItem;
    redQ2: TRichEdit;
    procedure mnuQuitClick(Sender: TObject);
    procedure mnuOptionAClick(Sender: TObject);
    procedure mnuOptionBClick(Sender: TObject);
    procedure mnuOptionCClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmQ2: TfrmQ2;
  DanceCouple : TDanceCouple;
  arrScores : TScoresArray;

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

```

procedure TfrmQ2.mnuOptionAClick(Sender: TObject);
var
  sPart1, sPart2 : String;
  cProf : char;
begin
  {Code Option A}
  sPart1 := InputBox('Partner 1', 'Enter the name of partner 1', 'Sarah');
  sPart2 := InputBox('Partner 2', 'Enter the name of partner 2', 'John');
  cProf := InputBox('Dance status', 'Enter the professional dance status of
the couple [A, B, C, D]', 'C')[1];
  DanceCouple := TDanceCouple.Create(sPart1, sPart2, cProf);
  redQ2.Lines.Add(DanceCouple.toString);
end;

procedure TfrmQ2.mnuOptionBClick(Sender: TObject);
var
  k : integer;
begin
  {Code Option B}
  For k := 1 to 4 do
    begin
      arrScores[k] := StrToInt(InputBox('Scores', 'Enter the score of judge' +
IntToStr(k), ''));
      if (arrScores[k] > 10) OR (arrScores[k] < 0)
        then arrScores[k] := 0;
    end;
  end;
  redQ2.Lines.Clear;
  redQ2.Lines.Add('Couple: ' + DanceCouple.GetDancePartner1 + ' & ' +
DanceCouple.GetDancePartner2 );
  redQ2.Lines.Add('Scores from the judges: ' + IntToStr(arrScores [1]) + ' ' +
IntToStr(arrScores [2]) + ' ' +IntToStr(arrScores [3]) + ' ' +
IntToStr(arrScores [4]) );
  redQ2.Lines.Add('Weighting value: ' + IntToStr(DanceCouple.GetWeighting));
  redQ2.Lines.Add('Final score: ' +
IntToStr(DanceCouple.CalcFinalScore(arrScores)));
end;

procedure TfrmQ2.mnuOptionCClick(Sender: TObject);
var
  TFile : TextFile;
  sLine : String;
begin
  {Code Option C}
  redQ2.Lines.Clear;
  AssignFile(TFile, 'Score.txt');
  Rewrite(TFile);
  sLine := DanceCouple.GetDancePartner1 + ', ' +
DanceCouple.GetDancePartner2 + ', ' +
IntToStr(DanceCouple.CalcFinalScore(arrScores));
  Writeln(TFile, sLine);
  CloseFile(TFile);
  redQ2.Lines.Add('Data has been written to the file');
end;

procedure TfrmQ2.mnuQuitClick(Sender: TObject);
begin
  Application.Terminate;
end;

end.

```

ANNEXURE F: SOLUTION with OOP – QUESTION 3: DELPHI

3.1. COUPLE CLASS UNIT:

```
unit uCouple;
//A solution for Question 3 - WITH OOP >> A Couple class
interface
type
    TCouple = class(TObject)
    private
        fNumber,
        fVotes      : integer;
        fEliminated : boolean;
    public
        constructor Create(iNumber : integer);
        function GetNumber : integer;
        procedure AddAVote;
        function GetVotes : integer;
        function GetEliminated : boolean;
        procedure SetEliminated(bEliminate : boolean);
        function toString : String;
    end;

implementation

uses SysUtils;

{ TCouple }

procedure TCouple.AddAVote;
begin
    Inc(fVotes , 1);
end;

constructor TCouple.Create(iNumber: integer);
begin
    fNumber := iNumber;
    fVotes  := 0;
    fEliminated := False;
end;

function TCouple.GetEliminated: boolean;
begin
    Result := fEliminated;
end;

procedure TCouple.SetEliminated(bEliminate: boolean);
begin
    fEliminated := bEliminate;
end;

function TCouple.GetNumber: integer;
begin
    Result := fNumber;
end;

function TCouple.GetVotes: integer;
begin
    Result := fVotes;
end;
```

```
function TCouple.toString: String;
begin
    IF GetEliminated
    then Result := IntToStr(fNumber) + #9 + '(Eliminated)'
    else Result := IntToStr(fNumber) + #9 + IntToStr(fVotes);
end;

end.
```

3.2 A VOTER CLASS UNIT:

```
unit uVoter;
// A solution for Question 3 - WITH OOP >> A Voter class
interface

type
    TVoter = class(TObject)
    private
        fName,
        fContact      : String;
        fAge           : integer;
    public
        constructor Create(sName, sContact : String; iAge : integer);
        function GetAge      : integer;
        function isInternational:boolean;
        function toString   : String;
    end;

implementation

{ TVoter }

constructor TVoter.Create(sName, sContact: String; iAge : integer);
begin
    fName      := sName;
    fContact   := sContact;
    fAge       := iAge;

end;

function TVoter.GetAge: integer;
begin
    Result := fAge;
end;

function TVoter.isInternational:boolean;
begin
    Result := (Copy(fContact, 1, 3) <> '+27');
end;

function TVoter.toString: String;
begin
    IF isInternational
    then Result := fName + #9 + fContact + #9 + 'International vote'
    else Result := fName + #9 + fContact;
end;

end.
```

3.3. FORM UNIT:

```
unit Question3U_OOP_MEMO;
// A solution for Question 3 with OOP
interface

uses
    Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
    Dialogs, Menus, StdCtrls, ComCtrls,
    uVoter, uCouple; //user created classes

type
    TfrmQ3OOPMemo = class(TForm)
        redQ3: TRichEdit;
        mmuMain: TMainMenu;
        mnuOptionA: TMenuItem;
        mnuOptionB: TMenuItem;
        mnuQuit: TMenuItem;
        procedure mnuQuitClick(Sender: TObject);
        procedure mnuOptionAClick(Sender: TObject);
        procedure mnuOptionBClick(Sender: TObject);
    private
        { Private declarations }
    public
        { Public declarations }
    end;

var
    frmQ3OOPMemo: TfrmQ3OOPMemo;

implementation

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

var
    arrCouples : array[1..14] of TCouple;
    Eliminated : Set of 1..14;

procedure TfrmQ3OOPMemo.mnuQuitClick(Sender: TObject);
begin
    Application.Terminate;
end;

procedure TfrmQ3OOPMemo.mnuOptionAClick(Sender: TObject);
var
    AVoter : TVoter;
    TFile : TextFile;
    sLine, sName, sContact : String;
    iDataCount, iInvalidCount : integer;
    iAge, iCouple, iNumber, K : integer;
begin //menu Option A
    //create the 14 couples.
    for K := 1 to 14 do
        arrCouples[K] := TCouple.Create(K);

    //user input which couples were eliminated
    Eliminated := [];
    For K := 1 to 4 do
        begin
            try
                Repeat
```



```

        iNumber := StrToInt(InputBox('Eliminated Couple', 'Enter the number of
the couple (1..14)', '3'));
    Until (iNumber in [1..14]);
except
    on EConvertError do
    begin
        ShowMessage('Please enter a number.');
```

Exit;

end;

end; //try

Eliminated := Eliminated + [iNumber];

arrCouples[iNumber].SetEliminated(True);

end;

//Read data from txt file

AssignFile(TFile, 'DataQ3.txt');

Reset(TFile);

iDataCount := 0;

iInvalidCount := 0;

redQ3.Lines.Clear;

redQ3.Paragraph.TabCount := 3;

redQ3.Paragraph.Tab[0] := 40;

redQ3.Paragraph.Tab[1] := 200;

redQ3.Paragraph.Tab[2] := 300;

redQ3.Lines.Add('No.'+#9+'Name'+#9+'Contact number');

While NOT EOF(TFile) DO

begin

Readln(TFile, sLine);

//split data >> name age contact & couple number

sName := Copy(sLine, 1, Pos(';', sLine)-1);

Delete(sLine, 1, Pos(';', sLine));

iAge := StrToInt(Copy(sLine, 1, Pos('#', sLine)-1));

Delete(sLine, 1, Pos('#', sLine));

sContact := Copy(sLine, 1, Pos(';', sLine)-1);

Delete(sLine, 1, Pos(';', sLine));

iCouple := StrToInt(sLine);

AVoter := TVoter.Create(sName, sContact, iAge);

IF (AVoter.GetAge >= 18) and NOT(iCouple IN Eliminated)

then

begin

Inc(iDataCount, 1);

redQ3.Lines.Add(IntToStr(iDataCount) + #9 + AVoter.toString);

arrCouples[iCouple].AddAVote;

end

else

begin

Inc(iInvalidCount, 1);

end;

end; //while

CloseFile(TFile);

redQ3.Lines.Add(' ');

redQ3.Lines.Add('Invalid votes: ' + IntToStr(iInvalidCount) + #13 +

'Valid votes: ' + IntToStr(iDataCount));

AVoter := nil;

end;

procedure TfrmQ3OOPMemo.mnuOptionBClick(Sender: TObject);

var

sLine : String;

K, iLow : integer;

```
begin
// Option B
redQ3.Lines.Clear;
redQ3.Paragraph.TabCount := 3;
redQ3.Paragraph.Tab[0] := 100;
redQ3.Paragraph.Tab[1] := 200;
redQ3.Paragraph.Tab[2] := 300;
sLine := '';
for K := 1 to 14 do
  IF arrCouples[K].GetEliminated
  then sLine := sLine + IntToStr(arrCouples[K].GetNumber) + ' ';

redQ3.Lines.Add('Votes received during week 5:');
redQ3.Lines.Add('Couple' + #9 + 'Votes');
for K := 1 to 14 do
  redQ3.Lines.Add(arrCouples[K].toString);

//determine the lowest total votes
iLow := 500;
For K := 1 to 14 do
  IF (arrCouples[K].GetVotes < iLow) AND (arrCouples[K].GetVotes <> 0)
  then
  begin
    iLow := arrCouples[K].GetVotes;
  end;

redQ3.Lines.Add(' ');
sLine := '';
For K:= 1 to 14 do
  IF (arrCouples[K].GetVotes = iLow)
  then
    sLine := sLine + IntToStr(arrCouples[K].GetNumber) + ' ';

  redQ3.Lines.Add('Couple(s) who may be eliminated next: ' + sLine);
end;

end.
```

ANNEXURE G: SOLUTION without OOP– QUESTION 3: DELPHI

```
unit Question3U_MEMO;
//A solution for question 3 without using OOP.
interface

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

type
    TfrmQuest3 = class(TForm)
        mnuMain: TMainMenu;
        mnuOptionA: TMenuItem;
        mnuOptionB: TMenuItem;
        mnuQuit: TMenuItem;
        redQ3: TRichEdit;
        procedure mnuQuitClick(Sender: TObject);
        procedure mnuOptionAClick(Sender: TObject);
        procedure mnuOptionBClick(Sender: TObject);
    private
        { Private declarations }
    public
        { Public declarations }
    end;

var
    frmQuest3: TfrmQuest3;

implementation

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

VAR
    iDataCount,
    iInvalidCount : integer;
    arrCouples    : array[1..14] of integer;
    Eliminated    : Set of 1..14;

procedure TfrmQuest3.mnuOptionAClick(Sender: TObject);
var
    TFile          : TextFile;
    sLine, sName, sContact : String;
    iAge, iCouple, iNumber, K : integer;
begin
    //user input - eliminated couples
    Eliminated := [];
    For K := 1 to 14 do
        arrCouples[K] := 0;
    For K := 1 to 4 do
        begin
            try
                Repeat
                    iNumber := StrToInt(InputBox('Eliminated couple', 'Enter the number of
the couple (1 - 14)', '3'));
                Until (iNumber in [1..14]);
            except
                on EConvertError do
                    begin
                        ShowMessage('Please type in a number.');

```

```

        end;
    end; //try
    Eliminated := Eliminated + [iNumber];
end;

//Read data from text file
AssignFile(TFile, 'DataQ3.txt');
Reset(TFile);
iDataCount := 0;
iInvalidCount := 0;
redQ3.Lines.Clear;
redQ3.Paragraph.TabCount := 3;
redQ3.Paragraph.Tab[0] := 40;
redQ3.Paragraph.Tab[1] := 200;
redQ3.Paragraph.Tab[2] := 300;
redQ3.Lines.Add('No.'+#9+ 'Name '+#9+ 'Cellphone number');

While NOT EOF(TFile) DO
begin
    Readln(TFile, sLine); {x}
    //split data >> name age contact & couple number
    sName := Copy(sLine, 1, Pos(';', sLine)-1);
    Delete(sLine, 1, Pos(';', sLine));
    iAge := StrToInt(Copy(sLine, 1, Pos('#', sLine)-1));
    Delete(sLine, 1, Pos('#', sLine));
    sContact := Copy(sLine, 1, Pos(';', sLine)-1);
    Delete(sLine, 1, Pos(';', sLine));
    iCouple := StrToInt(sLine);

    //validate >> Age>=18 couple not eliminated
    if (iAge >= 18) AND NOT(iCouple IN Eliminated)
    then
        begin //valid vote
            Inc(iDataCount, 1);
            Inc(arrCouples[iCouple], 1);

            sLine := IntToStr(iDataCount) + #9 + sName + #9 + sContact;
            if Copy(sContact, 1, 3) <> '+27' //International vote
            then
                sLine := sLine + #9 + 'International vote';

            redQ3.Lines.Add(sLine);
        end
    else
        begin
            //Invalid vote >> discard vote.
            Inc(iInvalidCount, 1);
        end; //else
    end; //while
    CloseFile(TFile);
    redQ3.Lines.Add(' ');
    redQ3.Lines.Add('Invalid votes: ' + IntToStr(iInvalidCount) + #13 +
        'Valid votes: ' + IntToStr(iDataCount));
end;

procedure TfrmQuest3.mnuOptionBClick(Sender: TObject);
var
    K, iCLow : integer;
    sLine : String;
begin
    //Option B
    redQ3.Lines.Clear;

```

```

redQ3.Paragraph.TabCount := 3;
redQ3.Paragraph.Tab[0] := 100;
redQ3.Paragraph.Tab[1] := 200;
redQ3.Paragraph.Tab[2] := 300;
sLine := '';

redQ3.Lines.Add('Votes received during week 5:');
redQ3.Lines.Add('Couple' + #9 + 'Votes');
for K := 1 to 14 do
begin
    if K in Eliminated
    then redQ3.Lines.Add(IntToStr(K) + #9 + '(Eliminated)')
    else redQ3.Lines.Add(IntToStr(K) + #9 + IntToStr(arrCouples[K]));
end;

//Calculate lowest number of votes
iCLow := 500;
For K := 1 to 14 do
    IF (arrCouples[K] < iCLow) AND (arrCouples[K] > 0)
    then iCLow := arrCouples[K];

//Display couples who may be eliminated next
redQ3.Lines.Add(' ');
sLine := '';
For K := 1 to 14 do
    IF (arrCouples[K] = iCLow)
    then
        sLine := sLine + IntToStr(K) + ' ';

redQ3.Lines.Add('Couple(s) who may be eliminated next: ' + sLine);
end;

procedure TfrmQuest3.mnuQuitClick(Sender: TObject);
begin
    Application.Terminate;
end;

end.
```

ANNEXURE H: SOLUTION – QUESTION 1: JAVA

```
// A solution for Question 1

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;
import java.sql.*;
import java.util.Scanner;

public class TestQuestion1
{
    public static void main(String[] args) throws SQLException, IOException
    {
        Scanner sc = new Scanner(System.in);
        // OR
        // BufferedReader inKb = new BufferedReader(new
        InputStreamReader(System.in));

        Question1 DB = new Question1();
        System.out.println();

        char choice = ' ';
        do {
            System.out.println("\n\n      MENU");
            System.out.println();
            System.out.println("      Option A");
            System.out.println("      Option B");
            System.out.println("      Option C");
            System.out.println("      Option D");
            System.out.println("      Option E");
            System.out.println("      Option F");
            System.out.println("      Option G");
            System.out.println();
            System.out.println("      Q - QUIT");
            System.out.println(" ");
            System.out.print("      Your choice? ");
            choice = sc.nextLine().toUpperCase().charAt(0);
            //OR
            //choice = inKb.readLine().toUpperCase().charAt(0);
            System.out.println(" ");
            String sql = "";
            switch (choice) {
                case 'A': // Question 1.1
                {
                    sql = "SELECT * FROM tblResults ORDER BY TypeOfDance,
RoutineNo Desc";
                    DB.query(sql);
                    break;
                }

                //=====
                case 'B': // Question 1.2
                {
                    sql = "SELECT RoutineNo, Week, TypeOfDance, Score FROM
tblResults WHERE (Score BETWEEN 25 AND 35) AND (Week=5 OR Week=9)";
                    DB.query(sql);
                    break;
                }
            }
        }

        //=====
    }
}
```

```

        case 'C': // Question 1.3
        {
            System.out.println("Question 1: Enter the TYPE of the
dance e.g. Rumba.");
            String sX = sc.nextLine();
                                //OR
                                // String sX = inKb.readLine();
            sql = "SELECT TypeOfDance, Count(*) AS
NumberOfPerformances FROM tblResults WHERE TypeOfDance = '" + sX + "' GROUP BY
TypeOfDance";

            DB.query(sql);
            break;
        }

//=====

        case 'D': // Question 1.4
        {
            sql = "SELECT Song, DancePartner1, DancePartner2 FROM
tblCouples, tblResults WHERE tblResults.DanceCoupleID=tblCouples.DanceCoupleID
AND (ProfessionalDancers = 'B') AND ((Song Like 'love%') OR (Song LIKE
'%you%'))";

            DB.query(sql);
            break;
        }

//=====

        case 'E': // Question 1.5
        {
            sql = "SELECT DanceCoupleID,
Format((Sum(Score)/Count(*)), '0.000') AS AverageScore FROM tblResults Group BY
DanceCoupleID";

            DB.query(sql);
            break;
        }

//=====

        case 'F': // Question 1.6
        {
            sql = "SELECT DancePartner1, DancePartner2 FROM
tblResults, tblCouples WHERE tblResults.DanceCoupleID=tblCouples.DanceCoupleID
AND (Result LIKE 'Eliminated') AND (Week < 12) GROUP BY DancePartner1,
DancePartner2";

            DB.query(sql);
            break;
        }

//=====

        case 'G': // Question 1.7
        {
            sql = "UPDATE tblResults SET Result = 'WINNERS' WHERE
Round = 2 AND DanceCoupleID = 8";
            DB.query(sql);
            break;
        }
    } while (choice != 'Q');
    DB.disconnect();
    System.out.println("Done");
}
}

```

ANNEXURE I: SOLUTION – QUESTION 2: JAVA

2.1 COUPLE OBJECT CLASS:

// POSSIBLE SOLUTION FOR QUESTION 2

```
public class DanceCouple
{
    private String dance_p1;
    private String dance_p2;
    private char professional;

    public DanceCouple()
    {
    }

    public DanceCouple(String part1, String part2, char pro)
    {
        dance_p1 = part1;
        dance_p2 = part2;
        professional = pro;
    }

    public String getDancePartner1()
    {
        return dance_p1;
    }

    public String getDancePartner2()
    {
        return dance_p2;
    }

    public char getProfessional()
    {
        return professional;
    }

    public int getWeighting()
    {
        int weighting = 0;
        if(professional == 'D')
            weighting = 3;

        if(professional == 'B' || professional == 'C')
        {
            weighting = 2;
        }
        if(professional == 'A')
        {
            weighting = 1;
        }
        return weighting;
    }

    public int calcFinalScore(int[] scores)
    {
        int result = 0;
        for(int i = 0; i < scores.length; i++)
        {
```



```

        if(i == 0 || i == 2)
        {
            result += scores[i] * getWeighting();
        }
        else
        {
            result += scores[i];
        }
    }
    return result;
}

public String toString()
{
    return "Couple: " + getDancePartner1()+ " & " + getDancePartner2()
+ "\n" + "Professional dance status: " + getProfessional();
}
}

```

2.3 TEST/DRIVER CLASS:

```

// POSSIBLE SOLUTION FOR QUESTION 2
import java.io.*;
import javax.swing.*;
import java.text.*;
import java.util.Scanner;

public class TestQuestion2
{
    public static void main(String[] args) throws Exception
    {
        int[] scores = new int[4];
        DanceCouple couple = new DanceCouple();

        Scanner sc = new Scanner(System.in);
        // OR
        // BufferedReader inKb = new BufferedReader(new
InputStreamReader(System.in));

        char choice = ' ';
        do {
            System.out.println("    MENU\n");
            System.out.println("Option A");
            System.out.println("Option B");
            System.out.println("Option C");
            System.out.println("");
            System.out.println("Q - QUIT");
            System.out.println("\nYour choice? ");
            choice = sc.nextLine().toUpperCase().charAt(0);
            //OR
            //choice = inKb.readLine().toUpperCase().charAt(0);
            switch (choice) {
                case 'A':
                    // OPTION A
                    System.out.println("Enter the name of partner 1 ");
                    String partner1 = sc.nextLine();
                    // OR
                    // String partner1 = inKb.readLine();
                    System.out.println("Enter the name of partner 2");
                    String partner2 = sc.nextLine();
                    // OR

```

```

        // String partner2 = inKb.readLine();
        System.out.println("Enter professional dance status of the
couple (A/B/C/D)");
        char professional = sc.nextLine().toUpperCase().charAt(0);
        // OR
        // char professional = inKb.readLine().toUpperCase().charAt(0);
        couple = new DanceCouple(partner1, partner2, professional);
        System.out.println(couple);
        System.out.println("");
        break;
    case 'B':
        // OPTION B
        System.out.println();
        for (int k = 0; k < 4; k++) {
            System.out.println("Enter the score from judge " + (k+1));
            scores[k] = Integer.parseInt(sc.nextLine());
            // OR
            // scores[k] = Integer.parseInt(inKb.readLine());
            if ((scores[k]>10)|| (scores[k]<0))
            {
                scores[k]=0;
            }
        }
        System.out.println("Couple: " + couple.getDancePartner1() + "
& " + couple.getDancePartner2());
        System.out.println("Scores from the judges: " + scores[0] + "
" + scores[1] + " " + scores[2] + " " + scores[3]);
        System.out.println("Weighting value: " +
couple.getWeighting());
        System.out.println("Final score: " +
couple.calcFinalScore(scores));
        System.out.println("");
        break;
    case 'C':
        // OPTION C
        BufferedWriter bw = new BufferedWriter(new
FileWriter("Score.txt"));
        bw.write(couple.getDancePartner1() + "," +
couple.getDancePartner2() + "," + couple.calcFinalScore(scores));
        bw.close();
        System.out.println("Data has been written to the text file");
        System.out.println("");
        break;
    case 'Q':
        System.out.println("QUIT");
    }
} while (choice != 'Q');
}
}

```

ANNEXURE J: SOLUTION with OOP – QUESTION 3: JAVA

3.1. COUPLE OBJECT CLASS:

//POSSIBLE SOLUTION FOR QUESTION 3 WITH OOP

```
public class Couple {

    private int num;
    private int numVotes;
    private boolean inComp; // couple still in competition

    public Couple(int num, int numVotes) {
        this.num = num;
        this.numVotes = numVotes;
        inComp = true;
    }

    public int getNum() {
        return num;
    }

    public void setNum(int num) {
        this.num = num;
    }

    public int getNumVotes() {
        return numVotes;
    }

    public void setNumVotes(int numVotes) {
        this.numVotes = numVotes;
    }

    public void addVote()
    {
        numVotes++;
    }

    public void setInComp()
    {
        inComp = false;
    }

    public boolean isInComp()
    {
        return inComp;
    }

    public String toString()
    {
        String output = "";
        if(inComp)
        {
            output = String.format("%-15s%-10s", getNum(),getNumVotes());
        }
        else
        {
            output = String.format("%-15s%-10s", getNum() ,"(Eliminated)");
        }
    }
}
```

```
        return output;
    }
}
```

3.2 A VOTER OBJECT CLASS:

```
public class Voter
{
    private String name;
    private String cellphone;
    private int age;
    private int vote;

    public Voter(String Name, String Cell, int Age, int Vote)
    {
        name = Name;
        cellphone = Cell;
        age = Age;
        vote = Vote;
    }

    public String getName() {
        return name;
    }

    public String getCellphone() {
        return cellphone;
    }

    public int getAge() {
        return age;
    }

    public int getVote() {
        return vote;
    }

    public boolean isInternational()
    {
        if(cellphone.substring(0,3).equals("+27"))
            return true;
        else
            return false;
    }

    public String toString()
    {
        String afvString = "";
        if(isInternational()){
            afvString = String.format("%-25s%-25s",getName(),getCellphone());
        }
        else
        {
            afvString = String.format("%-25s%-25s%-
25s",getName(),getCellphone(), "International vote");
        }
        return afvString;
    }
}
```

3.3 VotingProcess OBJECT CLASS:

```

import java.io.*;
import java.util.Scanner;

public class VotingProcess
{
    Couple[] couple = new Couple[14];
    public void readCouples() {
        for (int cnt = 0; cnt < 14; cnt++) {
            couple[cnt] = new Couple((cnt+1), 0);
        }
    }

    public void readFromFile() throws IOException
    {
        Scanner sf = new Scanner(new FileReader("DataQ3.txt"));
        // OR
        // BufferedReader vFile = new BufferedReader(new
        // FileReader("DataQ3.txt"));

        Scanner sc = new Scanner(System.in);
        // OR
        // BufferedReader kb = new BufferedReader(new
        // InputStreamReader(System.in));

        for (int cnt = 0; cnt < 4; cnt++) {
            int num = 0;
            do
            {
                System.out.println("Enter the number of couple (1 - 14) ");
                try
                {
                    num = Integer.parseInt(sc.nextLine());
                    // OR
                    // num = Integer.parseInt(kb.readLine());
                }
                catch (NumberFormatException e)
                {
                    num = 0;
                    System.out.println("Not number");
                }
            } while (num < 1 || num > 14);
            couple[num-1].setInComp();
        }

        String afvString = String.format("%-10s%-25s%-25s", "No.", "Name",
        "Contact number");
        System.out.println(afvString);
        int validVotes = 0;
        int invalidVotes = 0;
        /* Needed when working with BufferedReader
        * try {
        * String line = vFile.readLine();
        * while (line != null) { */

        while (sf.hasNext()) {
            String line = sf.nextLine();
            String[] temp1 = line.split(";");
            String[] temp2 = temp1[1].split("#");
            String name = temp1[0];

```

```

        int age = Integer.parseInt(temp2[0]);
        String cell = temp2[1];
        int vote = Integer.parseInt(temp1[2]);
        Voter voter = new Voter(name, cell, age, vote);
        if (voter.getAge() >= 18) {
            boolean found = false;
            for (int k = 0; k < 14; k++)
            {
                if (voter.getVote() == couple[k].getNum() &&
!couple[k].isInComp())
                    found = true;
            }
            if (found == false)
            {
                validVotes++;
                System.out.println(validVotes + "\t" + voter);
                couple[vote-1].addVote(); // count individual couple votes
            }
            else {
                invalidVotes++;
            }
        }
        else
        {
            invalidVotes++;
        }
        // line = vFile.readLine();
    }

    /* needed when working with BufferedReader
    }
    catch (FileNotFoundException e) {
        System.out.println(e);
    }
    catch (Exception f) {
        System.out.println(f);
    }
    */
    System.out.println();
    System.out.println("Invalid entries: " + invalidVotes);
    System.out.println("Valid entries: " + validVotes);
    System.out.println("\n");
}

public void displayCouples(){
    System.out.println("\n\nVotes received during week 5");
    System.out.printf(String.format("%-15s%-10s\n", "Couple", "Votes"));
    String output = "";
    for (int cnt = 0; cnt < 14; cnt++) {
        System.out.println(couple[cnt]);
    }
    System.out.println(output);
    System.out.print("Couple(s) who may be eliminated next: ");
    int lowest = getLow();

    for (int cnt = 0; cnt < 14; cnt++) {
        if (lowest == couple[cnt].getNumVotes())
        {
            System.out.print(couple[cnt].getNum() + " ");
        }
    }
    System.out.println("\n");
}
}

```

```

    public int getLow() {
        int low = 500;
        for (int cnt = 0; cnt < 14; cnt++) {
            if (couple[cnt].getNumVotes() < low && couple[cnt].getNumVotes() > 0)
            {
                low = couple[cnt].getNumVotes();
            }
        }
        return low;
    }
}

```

3.4 TEST/DRIVER CLASS (VotingMenu):

//POSSIBLE SOLUTION FOR QUESTION 3 WITH OOP

```

import java.io.*;
import java.util.Scanner;

public class VotingMenu {

    public static void main(String[] args) throws IOException
    {
        VotingProcess vote = new VotingProcess();
        Scanner sc = new Scanner(System.in);
        // OR
        // BufferedReader kb = new BufferedReader(new
InputStreamReader(System.in));
        char choice = ' ';
        do {
            System.out.println("    MENU\n");
            System.out.println("Option A");
            System.out.println("Option B");
            System.out.println("");
            System.out.println("Q - QUIT");
            System.out.println("\nYour choice? ");

            choice = sc.nextLine().toUpperCase().charAt(0);
            //OR
            //choice = kb.readLine().toUpperCase().charAt(0);
            switch (choice) {
                case 'A':
                    // OPTION A CODE HERE
                    vote.readCouples();
                    vote.readFromFile();
                    break;
                case 'B':
                    // OPTION B CODE HERE
                    vote.displayCouples();
                    break;

                case 'Q':
                    System.out.println("QUIT");
            }
        } while (choice != 'Q');
    }
}

```

ANNEXURE K: SOLUTION without OOP – QUESTION 3: JAVA

```
// A solution for Question 3 WITHOUT using OOP

import java.io.*;
import java.util.Scanner;

public class VotingProcess
{
    int[] dancers = new int[14];
    int[] numVotes = new int[14];
    int[] elim = new int[4];

    Scanner sc = new Scanner(System.in);
    // OR
    // BufferedReader kb = new BufferedReader(new
    InputStreamReader(System.in));

    VotingProcess() throws IOException {
        menu();
    }

    public void menu() throws IOException {
        char choice = ' ';
        do {
            System.out.println("    MENU\n");
            System.out.println("Option A");
            System.out.println("Option B");
            System.out.println("");
            System.out.println("Q - QUIT");
            System.out.println("\nYour choice?  ");

            choice = sc.nextLine().toUpperCase().charAt(0);
            // OR
            // choice = kb.readLine().toUpperCase().charAt(0);
            switch (choice) {
                case 'A':
                    // OPTION A CODE HERE
                    readCouples();
                    readFromFile();
                    break;
                case 'B':
                    // OPTION B CODE HERE
                    displayCouples();
                    break;
                case 'Q':
                    System.out.println("QUIT");
            }
        } while (choice != 'Q');
    }

    public void readCouples() {
        for (int cnt = 0; cnt < 14; cnt++) {
            dancers[cnt] = cnt + 1;
            numVotes[cnt] = 0;
        }
    }

    public void readFromFile() throws IOException
    {
        Scanner sf = new Scanner(new FileReader("DataQ3.txt"));
        Scanner sc = new Scanner(System.in);
    }
}
```



```
// OR  BufferedReader vFile = new BufferedReader(new
FileReader("DataQ3.txt"));
// BufferedReader kb = new BufferedReader(new
InputStreamReader(System.in));
for (int cnt = 0; cnt < 4; cnt++) {
    int num = 0;
    do {
        System.out.println("Enter the number of couple (1 - 14) ");
        try
        {
            num = Integer.parseInt(sc.nextLine());
            // OR
            // num = Integer.parseInt(kb.readLine());
        }
        catch (NumberFormatException e)
        {
            num = 0;
            System.out.println("Not number");
        }
    } while (num < 1 || num > 14);
    elim[cnt] = num;
}

String afvString = String.format("%-10s%-25s%-25s", "No.", "Name",
"Cellphone number");
System.out.println(afvString);
int validVotes = 0;
int invalidVotes = 0;
// Needed when working with BufferedReader
// try {
// String line = vFile.readLine();
// while (line != null)
//     while (sf.hasNext()) {
//         String line = sf.nextLine();
//         String[] temp1 = line.split(";");
//         String[] temp2 = temp1[1].split("#");
//         String name = temp1[0];
//         int age = Integer.parseInt(temp2[0]);
//         String cell = temp2[1];
//         int vote = Integer.parseInt(temp1[2]);
//         if (age >= 18) {
//             boolean found = false;
//             for (int k = 0; k<4; k++)
//                 if (vote == elim[k])
//                     found = true;

//             if (found == false) {
//                 validVotes++;
//                 if (cell.substring(0,3).equals("+27")){
//                     afvString = String.format("%-10s%-25s%-
25s",validVotes,name,cell);
//                 }
//                 else
//                 {
//                     afvString = String.format("%-10s%-25s%-
25s",validVotes,name,cell, "International vote");
//                 }
//                 System.out.println(afvString);
//                 numVotes[vote - 1]++; // count individual couple votes
//             } else {
//                 numVotes[vote - 1] = 0;
//                 invalidVotes++;
//             }
//         }
//     }
// }
```

```

        }
    } else {
        invalidVotes++;
    }
    //line = vFile.readLine();
}
/* Needed when working with BufferedReader
} catch (FileNotFoundException e) {
    System.out.println(e);
} catch (Exception f) {
    System.out.println(f);
}*/
System.out.println();
System.out.println("Invalid entries: " + invalidVotes);
System.out.println("Valid entries: " + validVotes);
System.out.println("\n");
}
public void displayCouples(){
    System.out.println("\n\nVotes received during week 5");
    System.out.printf(String.format("%-15s%-10s\n", "Couple", "Votes"));
    String output = "";
    for (int cnt = 0; cnt < 14; cnt++) {
        if(numVotes[cnt] > 0)
        { output = output + String.format("%-15s%-10s", dancers[cnt],
numVotes[cnt]) + "\n";
        }
        else
        {
            output = output + String.format("%-15s%-10s",
dancers[cnt], "(Eliminated)") + "\n";
        }
    }
    System.out.println(output);
    System.out.print("Couple(s) who may be eliminated next: ");
    int lowest = getLow();
    for (int cnt = 0; cnt < 14; cnt++) {
        if (lowest == numVotes[cnt]) {
            System.out.print(dancers[cnt] + " ");
        }
    }
    System.out.println("\n");
}

public int getLow() {
    int low = 500;
    for (int cnt = 0; cnt < 14; cnt++) {
        if (numVotes[cnt] < low && numVotes[cnt] > 0) {
            low = numVotes[cnt];
        }
    }
    return low;
}

public static void main(String[] args) throws IOException {
    new VotingProcess();
}
}

```

TOTAL: 120