



basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

FEBRUARY/MARCH 2014

MEMORANDUM

MARKS: 120

This memorandum consists of 33 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 the markers are required to attend a rigorous standardisation meeting to ensure that the guidelines are consistently interpreted and applied in the marking of candidates' practical work.
- 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 candidates who provide an alternate correct solution to that given in the marking guidelines will be given full credit for the relevant question.
- **Annexures A, B and C** (pages 3-7) include the marking grid for each question for using either one of the two programming languages.
- **Annexures D, E, F and G** (pages 8-20) contain the solutions for Delphi for Questions 1 to 3 in programming code.
- **Annexures H, I, J and K** (pages 21-33) contain the solutions for Java for Questions 1 to 3 in programming code.
- Copies of Annexures A, B and C (pages 3-7) should be made for each candidate and completed during the marking session.

ANNEXURE A:

QUESTION 1: MARKING GRID - PROGRAMMING AND DATABASE

CENTRE NUMBER:		EXAMINATION NUMBER:	
QUESTION	DESCRIPTION	MAX. MARKS	CANDIDATE'S MARKS
1.1	Query: Correct fields (or *)✓; correct table✓; ORDER BY DESC ✓	3	
	SQL: SELECT * FROM tblRespondents ORDER BY QuestID DESC		
1.2	Query: Correct fields & table✓; WHERE DateSubmitted ✓; > #correct date# ✓	3	
	SQL: SELECT QuestID, DateSubmitted, StudentID FROM tblRespondents WHERE DateSubmitted > #2013/08/07# Alternative: SELECT QuestID, DateSubmitted, StudentID FROM tblRespondents WHERE Day(DateSubmitted) > 7		
1.3	Query: Correct fields & table✓; WHERE City ✓; LIKE ✓; variable ✓; wildcards ✓; Number of devices >= 2 ✓; Internet contract ✓	7	
	SQL (D): SELECT City, NumMobileDevices, ConnectionType FROM tblRespondents WHERE City LIKE "%' + sX + '%" AND NumMobileDevices >= 2 AND InternetContract = TRUE SQL (J): SELECT City, NumMobileDevices, ConnectionType FROM tblRespondents WHERE City LIKE '%" + sX + '%'" AND NumMobileDevices >= 2 AND InternetContract = TRUE Alternative: yes/on/1 instead of true		
1.4	Query: Correct fields & table✓; Format & #0.00 ✓; Average ✓; AS AvgMobilePerCity ✓ GROUP BY City ✓	5	
	SQL (D): SELECT City, FORMAT(AVG(NumMobileDevices), "#0.00") AS AvgMobilePerCity FROM tblRespondents GROUP BY City Alternative: Format(AVG(NumMobileDevices), "#.00") Alternative: Format(AVG(NumMobileDevices), "0.00") Alternative: Format(AVG(NumMobileDevices), ".00") Alternative: Round(AVG(NumMobileDevices), 2) SQL (J): SELECT City, FORMAT(AVG(NumMobileDevices), '#0.00') AS AvgMobilePerCity FROM tblRespondents GROUP BY City Alternative: Format(AVG(NumMobileDevices), '#.00') Alternative: Format(AVG(NumMobileDevices), '0.00') Alternative: Format(AVG(NumMobileDevices), '.00') Alternative: ROUND(AVG(NumMobileDevices), 2)		

QUESTION 1: MARKING GRID - PROGRAMMING AND DATABASE – continued

1.5	Query: Correct fields✓; both tables✓; Count (any field)✓; AS NumQuestionnaires✓; WHERE clause linking both tables on StudentID✓; GROUP BY all fields✓	6	
	SQL: SELECT Name, Surname, YearOfStudy, COUNT(*) AS NumQuestionnaires FROM tblRespondents, tblStudents WHERE tblRespondents.StudentID = tblStudents.StudentID GROUP BY Name, Surname, YearOfStudy		
	Alternative:Use aliases for table names: SELECT Name, Surname, YearOfStudy, COUNT(*) AS NumQuestionnaires FROM tblRespondents R, tblStudents S WHERE R.StudentID = S.StudentID GROUP BY Name, Surname, YearOfStudy		
	Alternative:Use JOIN notation: SELECT Name, Surname, YearOfStudy, COUNT(*) AS NumQuestionnaires FROM tblRespondents INNER JOIN tblStudents ON tblRespondents.StudentID = tblStudents.StudentID GROUP BY Name, Surname, YearOfStudy		
	NOTE: May use LEFT JOIN or RIGHT JOIN as an alternative to INNER JOIN		
1.6	Query: UPDATE both tables✓; SET ✓; correct increment ✓; WHERE clause linking both tables on StudentID✓; AND correct name and surname✓	5	
	SQL (D): UPDATE tblRespondents, tblStudents SET NumMobileDevices = NumMobileDevices + 1 WHERE tblRespondents.StudentID = tblStudents.StudentID AND Name = "Kabelo" AND Surname = "Mkosi"		
	SQL (J): UPDATE tblRespondents, tblStudents SET NumMobileDevices = NumMobileDevices + 1 WHERE tblRespondents.StudentID = tblStudents.StudentID AND Name = 'Kabelo' AND Surname = 'Mkosi'		
1.7	Query: DELETE ✓; correct table ✓; WHERE no Internet contract✓ AND connectionType ✓; IS✓ NOT NULL ✓	6	
	SQL: DELETE FROM tblRespondents WHERE InternetContract = False AND ConnectionType IS NOT NULL		
	TOTAL:	35	

ANNEXURE B:

QUESTION 2: MARKING GRID - OBJECT-ORIENTED PROGRAMMING

CENTRE NUMBER:		EXAMINATION NUMBER:	
QUESTION	DESCRIPTION	MAX. MARKS	CANDIDATE'S MARKS
2.1.1	PARAMETERISED CONSTRUCTOR: Correct order✓ and data type of parameters✓; Assign four parameters✓	3	
2.1.2	calcAvg METHOD: Divide completed number of questionnaires by hours✓; Return floating point value ✓	2	
2.1.3	toString METHOD: Labels✓; <eoln> or #13 character ✓; Display all attributes correctly (character ✓) (numerical ✓) NOTE: May use private attributes/get methods	4	
2.2.1	INITIALISATION OF ARRAY: {DELPHI: AssignFile (1 mark), Reset (1 mark) JAVA: Create object to read from file (1 mark); instantiate object (1 mark) } ✓✓; Initialise loop counter ✓ Loop to read through file✓; Read a line from text file✓; Read next THREE lines from text file✓; Extract a character value from second line of text✓; Convert third line of text into an integer value ✓; Convert fourth line of text into a floating point value✓; Instantiate object using parameterized constructor: {object on left = ✓; class on right = ✓; parameters: type and order ✓} Change array counter✓; Close the text file✓	14	
2.2.2	MENU OPTION A: Heading ✓ Loop to read through array✓; Display data of objects from array using the toString() method✓	3	
2.2.3	MENU OPTION B: Initialize variables (averages of best male and female)✓ Loop to step through array ✓ Use calcAvg method ✓ IF male (M) ✓ AND calcAvg > highest male average✓ store name/position of highest✓ and replace highest male average with new highest value ✓ Repeat for female ✓ Output for best male and best female (name✓ and average✓); Format average to 2 decimal places✓	11	

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

2.2.4	MENU OPTION C: Input name, number of completed questionnaires and hours✓ Initialise flag ✓; Initialise counter ✓ Conditional loop (test array range✓ AND flag✓) IF name found ✓ change flag✓ set attributes at correct counter position in array using the set methods ✓; adding values typed in✓; make use of get method to retrieve previous value✓ Increment loop counter ✓ <i>Outside the loop:</i> Message if student not in list ✓	12	
	TOTAL:	49	

ANNEXURE C:

QUESTION 3: MARKING GRID – PROBLEM-SOLVING PROGRAMMING

CENTRE NUMBER:		EXAMINATION NUMBER:	
QUESTION	DESCRIPTION	MAX. MARKS	CANDIDATE'S MARKS
3.1	<p>Declare appropriate data structure (e.g. array) for unique names of games and a counter variable✓; Initialise array counter ✓</p> <p><i>Generate array with name of games:</i> Loop to step through array (arrData)✓ Extract the name of the game ✓(assign) ✓(copy/indexOf) ✓(use position of #) Initialise Boolean flag ✓ Loop to step through games array ✓ Test name of game✓ equals data array element ✓ If equal change Boolean flag ✓ Outside loop: If not found ✓ Increment the games array's counter ✓ Assign the name of game to games array ✓</p> <p><i>Displaying the names of the games:</i> Loop through games array ✓ Display menu repeatedly ✓ Display names of games ✓ from array ✓</p>	18	
3.2	<p>Accept user input (read from keyboard) ✓ Repeat✓ user input until valid input entered ✓</p> <p><i>Calculating statistics:</i> Declare and initialise counters for each device/total ✓ Identify correct game from array (arrGames) depending on user input ✓ Loop through array (arrData) ✓ Test if selected game is in arrData element ✓ Increment total counter✓; Test for device✓ as part of array element ✓; Increment correct device counter✓; Repeat for all three devices✓</p> <p><i>Display statistics:</i> Headings with the name of the selected game✓ and total number of times mentioned ✓; <i>Calculate percentage use of each device with formula:</i> Device counter/(total number of times mentioned)✓ * 100✓ Display use of each device rounded to one decimal✓ Display concatenated info in one line ✓</p>	18	
	TOTAL:	36	

SUMMARY OF CANDIDATE'S MARKS:

	QUESTION 1	QUESTION 2	QUESTION 3	GRAND TOTAL
MAX. MARKS	35	49	36	120
CANDIDATE'S MARKS				

ANNEXURE D: SOLUTION FOR 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)
    qryQ1: 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
  qryQ1.Close;
  qryQ1.SQL.Text := 'SELECT * FROM tblRespondents ORDER BY QuestID DESC';
  qryQ1.Open;
end;
//=====
procedure TfrmQ1.mnuOptionBClick(Sender: TObject);
begin
  qryQ1.Close;
  qryQ1.SQL.Text := 'SELECT QuestID, DateSubmitted, StudentID '+
    'FROM tblRespondents ' +
    'WHERE DateSubmitted > #2013/08/07#';

  qryQ1.Open;
end;
//=====

```



```

procedure TfrmQ1.mnuOptionCClick(Sender: TObject);
var
  sX : String;
begin
  sX := INPUTBOX('Question 1', 'Enter the name or part of the name of a city?',
    'town');

  qryQ1.Close;
  qryQ1.SQL.Text := 'SELECT City, NumMobileDevices, ConnectionType ' +
    'FROM tblRespondents ' +
    'WHERE City LIKE "%'+sX+'%" AND ' +
    'NumMobileDevices >= 2 AND InternetContract = TRUE';

  qryQ1.Open;
end;
//=====
procedure TfrmQ1.mnuOptionDClick(Sender: TObject);
begin
  qryQ1.Close;
  qryQ1.SQL.Text := 'SELECT City, FORMAT(AVG(NumMobileDevices),"0.00") ' +
    'AS AvgMobilePerCity ' +
    'FROM tblRespondents GROUP BY City';

  qryQ1.Open;
end;
//=====
procedure TfrmQ1.mnuOptionEClick(Sender: TObject);
begin
  qryQ1.Close;
  qryQ1.SQL.Text := 'SELECT Name, Surname, YearOfStudy, COUNT(*) AS
    NumQuestionnaires ' +
    'FROM tblRespondents R, tblStudents S ' +
    'WHERE (R.StudentID = S.StudentID) ' +
    'GROUP BY Name, Surname, YearOfStudy ';

  qryQ1.Open;
end;
//=====
procedure TfrmQ1.mnuOptionFClick(Sender: TObject);
begin
  qryQ1.Close;
  qryQ1.SQL.Text := 'UPDATE tblRespondents, tblStudents ' +
    'SET NumMobileDevices = NumMobileDevices + 1 ' +
    'WHERE tblRespondents.StudentID = tblStudents.StudentID ' +
    'AND Name = "Kabelo" AND Surname = "Mkosi"';

  qryQ1.ExecSQL;
  MessageDlg('Records Processed Successfully',mtInformation,[mbOk],0);

end;
//=====
procedure TfrmQ1.mnuOptionGClick(Sender: TObject);
begin
  qryQ1.Close;
  qryQ1.SQL.Text := 'DELETE FROM tblRespondents ' +
    'WHERE InternetContract = False AND ' +
    'ConnectionType IS NOT NULL';

  qryQ1.ExecSQL;
  MessageDlg('Records Processed Successfully',mtInformation,[mbOk],0);
end;
//=====
procedure TfrmQ1.mnuQuitClick(Sender: TObject);
begin
  Application.Terminate;
end;
end.

```

ANNEXURE E: SOLUTION FOR QUESTION 2: DELPHI

2.1. CLASS UNIT

```
unit uStudent_Memo;
//A solution for Question 2 - Class unit.
interface

TYPE
    TStudent = class(TObject)
    private
        fName           : String;
        fGender         : Char;
        fQuestionnaires : Integer;
        fHours          : Real;
    public
        constructor Create(sName:String; cGender:Char; iQuestionnaires:Integer;
rHours:Real);
        function calcAvg : Real;
        function toString : String;

        function GetName      : String;
        function GetGender    : Char;
        function GetQuestionnaires : Integer;
        procedure SetQuestionnaires (iQuestionnaires : Integer);
        function GetHours     : Real;
        procedure SetHours(rHours : Real);
    end;

implementation

uses SysUtils;

{ TStudent }

constructor TStudent.Create(sName: String; cGender: Char;
    iQuestionnaires: Integer; rHours: Real);
begin
    fName           := sName;
    fGender         := cGender;
    fQuestionnaires := iQuestionnaires;
    fHours          := rHours;
end;

function TStudent.calcAvg: Real;
begin
    Result := fQuestionnaires / fHours;
end;

function TStudent.toString: String;
begin
    Result := 'Student:' + fName + ' ('+fGender+')' + #13 +
        'Collected questionnaires: ' + IntToStr(fQuestionnaires) + #13 +
        'Total number of hours: ' + FloatToStr(fHours) + #13 +
        #13;
end;

function TStudent.GetName: String;
begin
    Result := fName;
end;
```

```
function TStudent.GetGender: Char;
begin
    Result := fGender;
end;

function TStudent.GetQuestionnaires: Integer;
begin
    Result := fQuestionnaires;
end;

procedure TStudent.SetQuestionnaires(iQuestionnaires: Integer);
begin
    fQuestionnaires := iQuestionnaires;
end;

function TStudent.GetHours: Real;
begin
    Result := fHours;
end;

procedure TStudent.SetHours(rHours: Real);
begin
    fHours := rHours;
end;

end.
```

2.2. MAIN FORM UNIT – QUESTION 2

```
unit Question2U_Memo;
//A solution for Question 2 - Main Form Unit.
interface

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

type
    TfrmQ2 = class(TForm)
        mnuMain: TMainMenu;
        mnuOptionA: TMenuItem;
        mnuQuit: TMenuItem;
        redQ2: TRichEdit;
        mnuOptionB: TMenuItem;
        mnuOptionC: TMenuItem;
        procedure mnuQuitClick(Sender: TObject);
        procedure mnuOptionAClick(Sender: TObject);
        procedure mnuOptionBClick(Sender: TObject);
        procedure FormCreate(Sender: TObject);
        procedure mnuOptionCClick(Sender: TObject);

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

var
    frmQ2: TfrmQ2;
    arrData : array[1..20] of TStudent;
    iCounter : Integer;
```

implementation

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

```
procedure TfrmQ2.FormCreate(Sender: TObject);
var
  TFile          : TextFile;
  iCollectQ      : Integer;
  sName, sGender : String;
  cGender        : Char;
  rHours         : Real;
begin
  IF NOT FileExists('DataQ2.txt') then
  begin
    MessageDlg('File does not exists.', mtInformation, [mbOK], 0);
    mnuOptionA.Visible := False;
    mnuOptionB.Visible := False;
    mnuOptionC.Visible := False;
    Exit;
  end;
  AssignFile(TFile, 'DataQ2.txt');
  Reset(TFile);
  iCounter := 0;
  while NOT EOF(TFile) Do
  begin
    Readln(TFile, sName);
    Readln(TFile, sGender);
    Readln(TFile, iCollectQ);
    Readln(TFile, rHours);
    cGender := sGender[1];
    Inc(iCounter, 1);
    arrData[iCounter] := TStudent.Create(sName, cGender, iCollectQ, rHours);
  end;
  CloseFile(TFile);
end;
```

```
procedure TfrmQ2.mnuOptionAClick(Sender: TObject);
var
  A : Integer;
begin
  // Menu Option A
  redQ2.Lines.Clear;
  redQ2.Lines.Add('List of students' + #13);
  for A := 1 to iCounter do
    redQ2.Lines.Add(arrData[A].toString);
end;
```

```
procedure TfrmQ2.mnuOptionBClick(Sender: TObject);
var
  A          : Integer;
  rHighM, rHighF : Real;
  sNameM, sNameF : String;
begin
  // Menu Option B
  redQ2.Lines.Clear;
  rHighM := 0;
  rHighF := 0;
  for A := 1 to iCounter do
  begin
    case arrData[A].GetGender of
      'M' : begin
```

```

        IF arrData[A].calcAvg > rHighM then
            begin
                rHighM := arrData[A].calcAvg;
                sNameM := arrData[A].GetName;
            end;
        end; //m.
    'F' : begin
        IF arrData[A].calcAvg > rHighF then
            begin
                rHighF := arrData[A].calcAvg;
                sNameF := arrData[A].GetName;
            end;
        end; //if
    end; //case
end; //for
redQ2.Lines.Add('Students with the highest average values: '+#13);
redQ2.Lines.Add('Male: ' + sNameM + ' with an average of ' +
    FloatToStrF(rHighM, ffFixed, 8,2));
redQ2.Lines.Add(' ');
redQ2.Lines.Add('Female: ' + sNameF + ' with an average of ' +
    FloatToStrF(rHighF, ffFixed, 8,2));
end;

procedure TfrmQ2.mnuOptionCClick(Sender: TObject);
var
    bFound          : Boolean;
    A, iQuestionnaires : Integer;
    rHours           : Real;
    sName            : String;
begin
    // Menu Option C
    sName := InputBox('Question 2', 'Name of student', 'Eliana');
    iQuestionnaires := StrToInt(InputBox('Question 2', 'Number of completed
questionnaires collected', '17'));
    rHours := StrToFloat(InputBox('Question 2', 'Number of hours', '1.5'));
    A := 1 ;
    bFound := False;
    while (A <= iCounter) AND NOT bFound do
        begin
            IF arrData[A].GetName = sName then
                begin
                    bFound := True;
                    arrData[A].SetQuestionnaires(arrData[A].GetQuestionnaires +
                        iQuestionnaires);
                    arrData[A].SetHours(arrData[A].GetHours + rHours);
                end
            else
                inc(A, 1);
            end; //while
        IF NOT bFound then
            begin
                redQ2.Lines.Clear;
                redQ2.Lines.Add('The student is not on the list');
            end
        else
            mnuOptionA.Click;
        end;
end;

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

end.
```

ANNEXURE F: SOLUTION QUESTION 3: DELPHI (OOP)

3.1. PLAYER CLASS UNIT

```
// An OOP solution for Question 3
unit uPlayer;
interface

type
    TPlayer = class(TObject)
    private
        fGameName    : String;
        fDevice      : String;
    public
        constructor Create(sGName, sDevice : String);
        procedure SetGameName(sGName : String);
        function GetGameName : String;
        procedure SetDevice(sDevice : String);
        function GetDevice : String;
        function toString : String;
    end;

implementation

{ TPlayer}
Uses SysUtils;

constructor TPlayer.Create(sGName, sDevice: String);
begin
    fGameName    := sGName;
    fDevice      := sDevice;
end;

function TPlayer.GetGameName: String;
begin
    Result := fGameName;
end;

function TPlayer.GetDevice: String;
begin
    result := fDevice;
end;

procedure TPlayer.SetGameName(sGName: String);
begin
    fGameName := sGName;
end;

procedure TPlayer.SetDevice(sDevice: String);
begin
    fDevice := sDevice;
end;

function TPlayer.toString: String;
begin
    Result := 'Player{' + 'Game = ' + fGameName + ', device = ' + fDevice + '}';
end;

end.
```

3.2. MAIN FORM UNIT - QUESTION 3

```

unit Question3OOP_U;
//An OOP solution for Question 3
interface

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

type
    TfrmQ3memo = class(TForm)
        redQ3: TRichEdit;
        mmuMain: TMainMenu;
        mnuOptionA: TMenuItem;
        mnuQuit: TMenuItem;
        procedure mnuQuitClick(Sender: TObject);
        procedure mnuOptionAClick(Sender: TObject);
        procedure FormCreate(Sender: TObject);
    private
        { Private declarations }
    public
        { Public declarations }
    end;

var
    frmQ3memo: TfrmQ3memo;

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

uses uPlayer;

var
    arrData : array[1..35] of String =
        ('Civilisation#PS3', 'Command & Conquer#PC', 'Solitaire#Xbox',
        'Chess#PC', 'Tetris#PC', 'Chess#PC', 'Command & Conquer#PC',
        'Civilisation#PC', 'SimCity#PC', 'Tetris#PC', 'SimCity#PC',
        'Civilisation#PS3', 'Tetris#PS3', 'Command & Conquer#PS3',
        'SimCity#PC', 'Solitaire#PC', 'Sims#Xbox', 'SimCity#Xbox',
        'Command & Conquer#PC', 'Chess#PS3', 'Tetris#Xbox',
        'Civilisation#Xbox', 'SimCity#PS3', 'Solitaire#PC',
        'Sims#Xbox', 'Command & Conquer#PS3', 'Command & Conquer#PS3',
        'Civilisation#PS3', 'Civilisation#PS3', 'Command & Conquer#Xbox',
        'SimCity#PS3', 'Solitaire#PS3', 'Civilisation#Xbox',
        'Command & Conquer#PC', 'SimCity#PC');

    arrGames : array[1..20] of String;
    iGCounter : Integer;

    arrPlayers : array[1..35] of TPlayer;

procedure CreatePlayerObjects;
var
    A : Integer;
    sGName, sDevice : String;
begin
    for A := 1 to length(arrData) do
        begin
            sGName := copy(arrData[A], 1, Pos('#', arrData[A])-1);
            sDevice := copy(arrData[A], Pos('#', arrData[A])+1, 5);

```

```

    arrPlayers[A] := TPlayer.Create(sGName, sDevice);
end;
end;

procedure CreateGamesArray;
var
    A, B      : Integer;
    sGName    : String;
    bFound    : Boolean;
begin
    for A := 1 to length(arrGames) do
        arrGames[A] := '';

        iGCounter := 0;
        for A := 1 to length(arrData) do
            begin
                sGName := arrPlayers[A].GetGameName;
                bFound := False;
                B := 1;
                while (B < length(arrGames)) and NOT bFound do
                    begin
                        IF sGName = arrGames[B] then
                            bFound := True
                        else Inc(B, 1);
                    end; //while
                IF NOT bFound then
                    begin
                        Inc(iGCounter, 1);
                        arrGames[iGCounter] := sGName;
                    end; //if
                end; //for A
            end;
        end;
    end;

procedure processData(iGameNo:Integer;var iPC,iXbox,iPS3,iCount:Integer);
var
    A      : Integer;
    sGName : String;
begin
    iPC := 0;
    iXbox := 0;
    iPS3 := 0;
    iCount := 0;
    sGName := arrGames[iGameNo];
    for A := 1 to length(arrPlayers) do
        begin
            IF UpperCase(sGName) = UpperCase(arrPlayers[A].GetGameName) then
                begin
                    Inc(iCount, 1);
                    IF UpperCase(arrPlayers[A].getDevice) = 'PC' then inc(iPC, 1);
                    IF UpperCase(arrPlayers[A].getDevice) = 'PS3' then inc(iPS3, 1);
                    IF UpperCase(arrPlayers[A].getDevice) = 'XBOX' then inc(iXbox, 1);
                end; //if.
            end; //for A.
        end;
    end;

procedure TfrmQ3memo.mnuOptionAClick(Sender: TObject);
var
    sGName      : String;
    A, iGameNo, iPC, iXbox, iPS3, iCount : Integer;
begin
    redQ3.Lines.Clear;
    redQ3.Lines.Add('List of games:');

```



```

redQ3.Lines.Add(' ');

for A := 1 to iGCounter do
    redQ3.Lines.Add(IntToStr(A) + '. ' + arrGames[A]);

repeat
    iGameNo := StrToInt(InputBox('Question 3', 'Enter the number of a game from the
                                list', '1'));

    IF NOT(iGameNo IN [1..iGCounter]) then
        ShowMessage('Invalid input'); // Of MessageDialog ...
until iGameNo in [1..iGCounter];

sGName := arrGames[iGameNo];
processData(iGameNo,iPC,iXbox,iPS3,iCount);
redQ3.Lines.Clear;
redQ3.Paragraph.TabCount := 2;
redQ3.Paragraph.Tab[0] := 50;
redQ3.Paragraph.Tab[1] := 100;
redQ3.Lines.Add(sGName + ' was mentioned ' + IntToStr(iCount)+ ' times.');
```

```

redQ3.Lines.Add(' ');
redQ3.Lines.Add('Percentage use of devices:');
redQ3.Lines.Add('PS3' + #9 + 'Xbox' + #9 + 'PC');
```

```

redQ3.Lines.Add(FloatToStrF(iPS3/iCount*100, ffFixed, 5,1) + '%' + #9 +
                FloatToStrF(iXbox/iCount*100, ffFixed, 5,1) + '%' + #9 +
                FloatToStrF(iPC/iCount*100, ffFixed, 5,1) + '%');
```

```

end;

procedure TfrmQ3memo.mnuQuitClick(Sender: TObject);
begin
    Application.Terminate;
end;

procedure TfrmQ3memo.FormCreate(Sender: TObject);
begin
    CreatePlayerObjects;
    CreateGamesArray;
end;

end.
```

ANNEXURE G: SOLUTION FOR QUESTION 3: DELPHI (Without OOP)

```
unit Question3U_MEMO;
//A solution for Question 3 (without OOP).
interface

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

type
    TfrmQ3 = class(TForm)
        redQ3: TRichEdit;
        mnuMain: TMainMenu;
        mnuOptionA: TMenuItem;
        mnuQuit: TMenuItem;
        procedure mnuOptionAClick(Sender: TObject);
        procedure mnuQuitClick(Sender: TObject);
    private
        { Private declarations }
    public
        { Public declarations }
    end;

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

var
    frmQ3: TfrmQ3;
    arrData : array[1..35] of String =
        ('Civilisation#PS3', 'Command & Conquer#PC', 'Solitaire#Xbox',
        'Chess#PC', 'Tetris#PC', 'Chess#PC', 'Command & Conquer#PC',
        'Civilisation#PC', 'SimCity#PC', 'Tetris#PC', 'SimCity#PC',
        'Civilisation#PS3', 'Tetris#PS3', 'Command & Conquer#PS3',
        'SimCity#PC', 'Solitaire#PC', 'Sims#Xbox', 'SimCity#Xbox',
        'Command & Conquer#PC', 'Chess#PS3', 'Tetris#Xbox',
        'Civilisation#Xbox', 'SimCity#PS3', 'Solitaire#PC',
        'Sims#Xbox', 'Command & Conquer#PS3', 'Command & Conquer#PS3',
        'Civilisation#PS3', 'Civilisation#PS3', 'Command & Conquer#Xbox',
        'SimCity#PS3', 'Solitaire#PS3', 'Civilisation#Xbox',
        'Command & Conquer#PC', 'SimCity#PC');

    arrGames : array[1..20] of String;
    iGCounter : Integer;

procedure CreateGamesArray;
var
    A, B : Integer;
    bFound : Boolean;
    sGName : String;
begin
    for A := 1 to length(arrGames) do
        arrGames[A] := '';

    iGCounter := 0;
    for A := 1 to length(arrData) do
        begin
            sGName := Copy(arrData[A], 1, Pos('#', arrData[A])-1);
            bFound := False;
```

```

    B := 1;
    while (B < arrSpeletjies) AND NOT bFound do
        begin
            IF sGName = arrGames[B] then
                bFound := True
            else Inc(B, 1);
        end; //while
    IF NOT bFound then
        begin
            Inc(iGCounter, 1);
            arrGames[iGCounter] := sGName;
        end; //if
    end; //for A
end;

procedure ProcessData(iGameNo:Integer; var iPC, iXbox, iPS3, iCount:Integer);
var
    sGName : String;
    A      : Integer;
begin
    iPC      := 0;
    iXbox    := 0;
    iPS3     := 0;
    iCount   := 0;
    sGName := arrGames[iGameNo];
    for A := 1 to length(arrData) do
        begin
            IF Pos(sGName, arrData[A]) > 0 then
                begin
                    Inc(iCount, 1);
                    IF pos('#PC', arrData[A]) > 0 then inc(iPC, 1);
                    IF pos('#PS3', arrData[A]) > 0 then inc(iPS3, 1);
                    IF pos('#Xbox', arrData[A]) > 0 then inc(iXbox, 1);
                end; //if
            end; //for A
        end;
    end;

procedure TfrmQ3.mnuOptionAClick(Sender: TObject);
var
    sGName, sDevice : String;
    A, iGameNo, iPC, iXbox, iPS3, iCount : Integer;
begin
    CreateGamesArray;
    redQ3.Lines.Clear;
    redQ3.Lines.Add('List of Games');
    redQ3.Lines.Add(' ');
    for A := 1 to iGCounter do
        redQ3.Lines.Add(IntToStr(A) + '. ' + arrGames[A]);
    repeat
        iGameNo := StrToInt(InputBox('Question 3', 'Enter the number of a games from the list', '1'));
    until iGameNo in [1..iGCounter];

    IF NOT(iGameNo in [1..iGCounter]) then
        ShowMessage('Invalid input');
    until iGameNo in [1..iGCounter];

    sGName := arrGames[iGameNo];

    ProcessData(iGameNo, iPC, iXbox, iPS3, iCount);
    redQ3.Lines.Clear;

```

```
redQ3.Paragraph.TabCount := 2;
redQ3.Paragraph.Tab[0] := 50;
redQ3.Paragraph.Tab[1] := 100;

redQ3.Lines.Add(sGName + ' was mentioned ' + IntToStr(iCount)+ ' times.');
```

```
redQ3.Lines.Add(' ');
redQ3.Lines.Add('Percentage use of devices:');
redQ3.Lines.Add('PS3' + #9 + 'Xbox' + #9 + 'PC');
```

```
redQ3.Lines.Add(FloatToStrF(iPS3/iCount*100, ffFixed, 5,1) + '%' + #9 +
                  FloatToStrF(iXbox/iCount*100, ffFixed, 5,1) + '%' + #9 +
                  FloatToStrF(iPC/iCount*100, ffFixed, 5,1) + '%');
```

```
end;
```

```
procedure TfrmQ3.mnuQuitClick(Sender: TObject);
begin
    Application.Terminate;
end;
```

```
end.
```

ANNEXURE H: SOLUTION FOR QUESTION 1: JAVA

```
//A solution for Question 1
import java.io.*;
import java.sql.*;
import javax.swing.*;
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 tblRespondents ORDER BY QuestID DESC";
                    DB.query(sql);
                    break;
                }
//=====
                case 'B':          // Question 1.2
                {
                    sql = "SELECT QuestID, DateSubmitted, StudentID FROM tblRespondents
                        WHERE DateSubmitted > #2013/08/07#";
                    DB.query(sql);
                    break;
                }
//=====
            }

```

```

        case 'C': // Question 1.3
        {
            System.out.println("Enter the name or part of the name of a city:");
            String sX = sc.nextLine();
            // OR String sX = inKb.readLine();
            sql = "SELECT City, NumMobileDevices, ConnectionType FROM
                tblRespondents WHERE City LIKE '%" + sX + "%' AND
                NumMobileDevices >= 2 AND InternetContract = TRUE";
            DB.query(sql);
            break;
        }
//=====
        case 'D': // Question 1.4
        {
            sql = "SELECT City, FORMAT(AVG(NumMobileDevices), '0.00') AS
                AvgMobilePerCity FROM tblRespondents GROUP BY City";
            DB.query(sql);
            break;
        }
//=====
        case 'E': // Question 1.5
        {
            sql = "SELECT Name, Surname, YearOfStudy,COUNT(*) AS NumQuestionnaires
                FROM tblRespondents, tblStudents WHERE tblRespondents.StudentID =
                tblStudents.StudentID GROUP BY Name, Surname, YearOfStudy";
            DB.query(sql);
            break;
        }
//=====
        case 'F': // Question 1.6
        {
            sql = "UPDATE tblRespondents,tblStudents SET NumMobileDevices =
                NumMobileDevices + 1 WHERE tblRespondents.studentID =
                tblStudents.studentID AND name = 'Kabelo' AND surname = 'Mkosi'";
            DB.query(sql);
            break;
        }
//=====
        case 'G': // Question 1.7
        {
            sql = "DELETE FROM tblRespondents WHERE InternetContract = False AND
                ConnectionType IS NOT NULL";
            DB.query(sql);
            break;
        }
    }
    }while (choice != 'Q');
    DB.disconnect();
    System.out.println("Done");
}
}

```

ANNEXURE I: SOLUTION FOR QUESTION 2: JAVA

2.1. OBJECT CLASS

```
//A solution for Question 2 - OOP

public class StudentMemo {

    private String name;
    private char gender;
    private int questionnaires;
    private double hours;

    public StudentMemo( String name,  char gender, int questionnaires, double hours) {
        this.name = name;
        this.gender = gender;
        this.questionnaires = questionnaires;
        this.hours = hours;
    }

    public double calcAve() {
        return questionnaires / hours;
    }

    public String toString() {
        return "Student: " + getName() + " (" + gender + ")\nCollected
questionnaires: " + getQuestionnaires() + "\nTotal number of hours: " + getHours() +
"\n";
    }

    public String getName() {
        return name;
    }

    public char getGender() {
        return gender;
    }

    public int getQuestionnaires() {
        return questionnaires;
    }

    public void setQuestionnaires(int questionnaires) {
        this.questionnaires = questionnaires;
    }

    public double getHours() {
        return hours;
    }

    public void setHours(double hours) {
        this.hours = hours;
    }
}
```

2.2. TEST/DRIVER CLASS - QUESTION 2

```
// Asolution for Question 2 - OOP
import java.io.*;
import java.text.DecimalFormat;
import java.util.Scanner;

public class TestQuestion2_Memo {

    public static void main(String[] args) throws IOException {
        Scanner sc = new Scanner(System.in);
        Scanner sf;
        // OR BufferedReader bf;
        //BufferedReader kb = new BufferedReader(new InputStreamReader(System.in));

        int counter = 0;
        StudentMemo[] arrData = new StudentMemo[20];

        // try {
            sf = new Scanner(new FileReader("DataQ2.txt"));

            // OR bf = new BufferedReader(new FileReader("DataQ2.txt"));
            // String name = bf.readLine();
            // while (name != null)
            while (sf.hasNext())
            {
                String name = sf.nextLine();
                char gender = sf.nextLine().charAt(0);
                // OR char gender = bf.readLine().charAt(0);
                int questionnaires = Integer.parseInt(sf.nextLine());
                // OR int questionnaires = Integer.parseInt(bf.readLine());
                double hours = Double.parseDouble(sf.nextLine());
                // OR double hours = Double.parseDouble( bf.readLine());
                arrData[counter] = new StudentMemo(name, gender, questionnaires, hours);
                counter++;
                // OR name = bf.readLine();
            }
            sf.close();
            /* Needed when using BufferedReader
                bf.close();
            } catch (FileNotFoundException e) {
                System.out.println("File does not exist");
                System.exit(0);
            } catch (Exception f) {
                System.out.println(f);
            } */

            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?  ");
            } while (choice != 'Q');
```



```

choice = sc.nextLine().toUpperCase().charAt(0);
// OR choice = kb.readLine().toUpperCase().charAt(0);
switch (choice) {
    case 'A':
        // display array using toString method
        System.out.println("List of students\n");
        for (int count = 0; count < counter; count++) {
            System.out.println(arrData[count]);
        }
        break;

    case 'B':
        DecimalFormat df = new DecimalFormat("0.00");
        String nameM = "";
        String nameF = "";
        double highM = 0;
        double highF = 0;
        for (int count = 0; count < counter; count++)
        {
            double avg = arrData[count].avgNum();
            if (arrData[count].getGender() == 'M' && avg > highM)
            {
                highM = avg;
                nameM = arrData[count].getName();
            }
            if (arrData[count].getGender() == 'F' && avg > highF)
            {
                highF = avg;
                nameF = arrData[count].getName();
            }
        }
        System.out.println("Students with the highest average values:\n");
        System.out.println("Male: " + nameM + " with an average of " +
                           df.format(highM) + "\n");
        System.out.println("Female: " + nameF + " with an average of " +
                           df.format(highF) + "\n");
        break;

    case 'C':
        System.out.println("Name of student: ");
        String name = sc.nextLine();
        // OR String name = kb.readLine();

        System.out.println("Number of completed questionnaires collected: ");
        String collectedQ = sc.nextLine();
        // OR String collectedQ = kb.readLine();

        System.out.println("Number of hours: ");
        String newhours = sc.nextLine();
        // OR String newhours = kb.readLine();

        boolean found = false;
        int count = 0;
        while (found == false && count < counter) {
            if (arrData[count].getName().equalsIgnoreCase(name)) {
                found = true;
                int questionnaires = arrData[count].getQuestionnaires();
                double hours = arrData[count].getHours();
                arrData[count].setQuestionnaires(questionnaires +
                                                    Integer.parseInt(collectedQ));
                arrData[count].setHours(hours + Double.parseDouble(newhours));
            }
        }
    
```

```
        else
            count++;
    }

    if (found == false) {
        System.out.println("The student is not on the list");
    }
    else
    {
        System.out.println(arrData[count].toString());
    }
    break;

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

ANNEXURE J: SOLUTION FOR QUESTION 3: JAVA (OOP)

3.1. PLAYER OBJECT CLASS:

```
//An OOP solution for Question 3
public class Player {
    private String game;
    private String device;

    public Player(String game, String device) {
        this.game = game;
        this.device = device;
    }

    public String getDevice() {
        return device;
    }

    public void setDevice(String device) {
        this.device = device;
    }

    public String getGameName() {
        return game;
    }

    public void setGameName(String game) {
        this.game = game;
    }

    public String toString() {
        return "Player{" + "game = " + game + ", device = " + device + "}";
    }
}
```

3.2. SURVEYSTATS OBJECT CLASS

```
//OOP solution for Question 3
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.text.DecimalFormat;
import java.util.Scanner;

public class SurveyStats {

    Scanner sc = new Scanner(System.in);
    // OR BufferedReader kb = new BufferedReader(new InputStreamReader(System.in));
    String[] arrGames = new String [20];
    int gamesCounter = 0;
    Player [] arrPlayers = new Player[35];

    public void populateArr(String [] arrData) {
        for (int cnt = 0; cnt < arrData.length; cnt++)
        {
            String[] arrItems = arrData[cnt].split("#");
            arrPlayers[cnt] = new Player(arrItems[0], arrItems[1]);
        }
    }
}
```

```
public void createGamesArray()
{
    String gameName = "";
    int b = 0;
    for (int cnt = 0; cnt < arrPlayers.length; cnt++)
    {
        gameName = arrPlayers[cnt].getGameName();
        boolean found = false;
        while (!(found) && (b < arrGames.length))
        {
            if ( gameName.equalsIgnoreCase(arrGames[b]))
                found = true;
            else
                b++;
        }
        if (!(found))
        {
            arrGames[gamesCounter] = gameName;
            gamesCounter++;
        }
        b = 0;
    }
    gamesCounter--;
}

public int getGamesCounter()
{
    return gamesCounter;
}

public int displayMenu() throws IOException {
    System.out.println("List of games:\n=====");
    int cnt;
    for (cnt = 0; cnt <= gamesCounter; cnt++) {
        System.out.println((cnt + 1) + "\t" + arrGames[cnt]);
    }
    int gameNr = 0;
    do
    {
        try
        {
            System.out.print("\nEnter the number of a game from the list > ");
            gameNr = Integer.parseInt(sc.nextLine());
            // OR int gameNr = Integer.parseInt(kb.readLine());
        } catch (NumberFormatException E)
        {
            System.out.println("Invalid input");
        }
    }while (!(gameNr > 0) && (gameNr <= gamesCounter+1));
    return gameNr;
}

public void getStats(int choice)
{
    String gameOfChoice = arrGames[choice];
    int cntPS3 = 0;
    int cntXbox = 0;
    int cntPC = 0;
    double number = 0;
```

```

        for (int cnt = 0; cnt < arrPlayers.length; cnt++) {
            if (arrPlayers[cnt].getGameName().equals(gameOfChoice)) {
                number++;
                String device = arrPlayers[cnt].getDevice();
                if (device.indexOf("PS3") >= 0) {
                    cntPS3++;
                }
                if (device.indexOf("Xbox") >= 0) {
                    cntXbox++;
                }
                if (device.indexOf("PC") >= 0) {
                    cntPC++;
                }
            }
        }
        System.out.println(gameOfChoice + " was mentioned " + number + " times." );
        System.out.println("");
        System.out.println("Percentage use of devices");
        String headings = String.format("%-20s%-20s%-20s", "PS3", "Xbox", "PC");
        String outString = String.format("%-3.1f%-16s%-3.1f%-16s%-3.1f%-16s", (cntPS3 / number * 100), "%", (cntXbox / number * 100), "%", (cntPC / number * 100), "%");
        System.out.println(headings);
        System.out.println(outString);
        System.out.println();
    }
}

```

3.3. TESTQUESTION3 DRIVER CLASS

```

//An OOP solution for Question 3
import java.io.*;
import java.util.Scanner;

public class TestQuestion3{
    public static void main(String[] args) throws IOException {
        Scanner sc = new Scanner(System.in);
        // OR BufferedReader bf = new BufferedReader(new
        InputStreamReader(System.in));

        String[] arrData = {"Civilisation#PS3", "Command & Conquer#PC",
        "Solitaire#Xbox",
        "Chess#PC", "Tetris#PC", "Chess#PC", "Command & Conquer#PC",
        "Civilisation#PC", "SimCity#PC", "Tetris#PC", "SimCity#PC",
        "Civilisation#PS3", "Tetris#PS3", "Command & Conquer#PS3",
        "SimCity#PC", "Solitaire#PC", "Sims#Xbox", "SimCity#Xbox",
        "Command & Conquer#PC", "Chess#PS3", "Tetris#Xbox",
        "Civilisation#Xbox", "SimCity#PS3", "Solitaire#PC", "Sims#Xbox",
        "Command & Conquer#PS3", "Command & Conquer#PS3", "Civilisation#PS3",
        "Civilisation#PS3", "Command & Conquer#Xbox", "SimCity#PS3",
        "Solitaire#PS3", "Civilisation#Xbox", "Command & Conquer#PC",
        "SimCity#PC"};

        SurveyStats obj = new SurveyStats();

        char cChoice = ' ';
        int gChoice = 0;

        obj.populateArr(arrData);
        obj.createGamesArray();
    }
}

```

```
int gamesCounter = obj.getGamesCounter();
do {
    System.out.println("=====");
    System.out.println("    MENU\n");
    System.out.println("Option A");
    System.out.println("");
    System.out.println("Q - QUIT");
    System.out.println("\nYour choice?  ");
    cChoice = sc.nextLine().toUpperCase().charAt(0);
    // OR cChoice = kb.readLine().toUpperCase().charAt(0);
    switch (cChoice) {
        case 'A':
            gChoice = obj.displayMenu();
            obj.getStats(gChoice-1);
            break;
        case 'Q':
            System.out.println("Quit");
            break;
    }
} while (cChoice != 'Q');
}
```

ANNEXURE K: SOLUTION FOR QUESTION 3: JAVA (Without OOP)

```
//A solution for Question 3 without OOP
import java.util.Scanner;
import java.text.*;
import java.io.*;
public class TestQuestion3
{
    public TestQuestion3()
    {
        CreateGamesArray();
        MainMenu();
    }

    public static void main(String[] args)
    {
        new TestQuestion3();
    }

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

    String[] arrData =
    {"Civilisation#PS3", "Command & Conquer#PC", "Solitaire#Xbox",
        "Chess#PC", "Tetris#PC", "Chess#PC", "Command & Conquer#PC",
        "Civilisation#PC", "SimCity#PC", "Tetris#PC", "SimCity#PC",
        "Civilisation#PS3", "Tetris#PS3", "Command & Conquer#PS3",
        "SimCity#PC", "Solitaire#PC", "Sims#Xbox", "SimCity#Xbox",
        "Command & Conquer#PC", "Chess#PS3", "Tetris#Xbox",
        "Civilisation#Xbox", "SimCity#PS3", "Solitaire#PC", "Sims#Xbox",
        "Command & Conquer#PS3", "Command & Conquer#PS3", "Civilisation#PS3",
        "Civilisation#PS3", "Command & Conquer#Xbox", "SimCity#PS3",
        "Solitaire#PS3", "Civilisation#Xbox", "Command & Conquer#PC",
        "SimCity#PC"};

    String[] arrGames = new String[20];
    int number = 0;

    public void CreateGamesArray()
    {
        for (int k = 0; k < arrData.length;k++)
        {
            boolean found = false;
            String[] temp = arrData[k].split("#");
            int test = 0;
            do
            {
                if (temp[0].equals(arrGames[test]))
                {
                    found = true;
                }
                else
                {
                    test++;
                }
            } while ((test < number) && (found==false));
        }
    }
}
```

```

        if (found == false)
        {
            arrGames[number] = temp[0];
            number++;
        }
    }
}

public void MainMenu()
{
    char choice;
    do {
        System.out.println("    MENU\n");
        System.out.println("Option A");
        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':
                System.out.println("List of games\n");
                for (int cnt = 0; cnt < number; cnt++)
                {
                    System.out.println((cnt+1) + ". " + arrGames[cnt]);
                }
                int gameNr = 0;
                do
                {
                    try
                    {
                        System.out.print("\nEnter the number of a game from the list > ");
                        gameNr = Integer.parseInt(sc.nextLine());
                        // OR gameNr = Integer.parseInt(bf.readLine());
                    }
                    catch (NumberFormatException E)
                    {
                        System.out.println("Invalid input");
                    }
                }while (!((gameNr > 0) && (gameNr < number)));

                if (gameNr < number + 1)
                {
                    String game = arrGames[gameNr-1];
                    int pc = 0;
                    int ps3 = 0;
                    int xBox = 0;

                    for (int k = 0; k < arrData.length; k++)
                    {
                        if (arrData[k].toUpperCase().indexOf(game.toUpperCase())== 0)
                        {
                            if (arrData[k].indexOf("PC")>0)
                            {
                                pc++;
                            }
                            if (arrData[k].indexOf("PS3")>0)
                            {
                                ps3++;
                            }
                        }
                    }
                }
            }
        }
    }
}

```



```

        if (arrData[k].indexOf("Xbox")>0)
        {
            xBox++;
        }
    }
    // output
    double tot = pc + ps3 + xBox;
    System.out.println("\n\n");
    System.out.println(game + " was mentioned " + tot + " times." );
    System.out.println("");
    System.out.println("Percentage use of devices:");
    String headings =String.format("%-20s%-20s%-20s", "PS3","Xbox", "PC");
    String outString = String.format("%-3.1f%-16s%-3.1f%-16s%-3.1f%-16s",
        s3/tot* 100), "%",(xBox /tot * 100), "%", (pc/tot * 100), "%");
    System.out.println(headings);
    System.out.println(outString);
    System.out.println("\n");
}
else
{
    System.out.println("Quit");
}

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

```