basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

FEBRUARY/MARCH 2011

MEMORANDUM

MARKS: 120

The memorandum consists of 27 pages.

GENERAL INFORMATION:

- **Pages 2 – 12 contain the Delphi memoranda of possible solutions for QUESTIONS 1 to 3 in programming code.**
- **Pages 13 – 21 contain the Java memoranda of possible solutions for QUESTIONS 1 to 3 in programming code.**
- **Pages 22 – 27 contain ANNEXURES A to F which include a marking grid for each question for candidates using either one of the two programming languages.**
- **Copies of the ANNEXURES should be made for each learner to be completed during the marking session.**

SECTION A: DELPHI

QUESTION 1: PROGRAMMING AND DATABASE (DELPHI)

```
unit Question1_U;

interface

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

type
  TfrmRec = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    btnA: TButton;
    btnB: TButton;
    btnC: TButton;
    btnD: TButton;
    btnE: TButton;
    btnF: TButton;
    btnG: TButton;
    BitBtn1: TBitBtn;
    qryRec: TADOQuery;
    tblRecAg: TDataSource;
    grdRec: TDBGrid;

    procedure btnAClick(Sender: TObject);
    procedure btnBClick(Sender: TObject);
    procedure btnCClick(Sender: TObject);
    procedure btnDClick(Sender: TObject);
    procedure btnEClick(Sender: TObject);
    procedure btnFClick(Sender: TObject);
    procedure btnGClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmRec: TfrmRec;

implementation

{$R *.dfm}
```

```
procedure TfrmRec.btnAClick(Sender: TObject);
begin
  qryRec.Active := False;                                    //Question 1.1

  qryRec.SQL.Text := 'SELECT *✓ FROM tblAgencies✓ ORDER BY NumPrevPlacements✓
DESC'✓;
  qryRec.Active := True;
end;                                                                    (4)
//=========================================================================
procedure TfrmRec.btnBClick(Sender: TObject);              //Question 1.2
begin
  qryRec.Active := False;
  qryRec.SQL.Text := 'SELECT Name, Surname, Salary✓ FROM tblClients✓ WHERE
                              FullTime = False✓ AND Salary < 15000✓';
  qryRec.Active := True;
end;                                                                    (4)
//=========================================================================
                                                           //Question 1.3
procedure TfrmRec.btnCClick(Sender: TObject);
begin
  qryRec.Active := False;
  qryRec.SQL.Text := 'SELECT Count(*)✓ AS [The number of agencies that offer
international jobs are] ✓ FROM tblAgencies✓ WHERE International=True✓';
  qryRec.Active := True;
end;                                                                    (4)
//=========================================================================
procedure TfrmRec.btnDClick(Sender: TObject);              //Question 1.4
begin
  qryRec.Active := False;
  qryRec.SQL.Text := 'SELECT Name,Surname,Salary✓,format✓ (salary * (10 /
                  100),".00"✓)AS [AgentComm] ✓ FROM tblClients✓';
  qryRec.Active := True;
end;                                          OR:(salary * 0.1, 2)
                                                                        (5)
//=========================================================================
procedure TfrmRec.btnEClick(Sender: TObject);              //Question 1.5
begin
  qryRec.Active := False;
  qryRec.SQL.Text :=  'INSERT INTO tblAgencies✓ VALUES✓ ("Jobs
                              Unlimited","Western Cape",False,0) ✓✓ ';

  qryRec.ExecSQL;
  MessageDlg ( 'Record inserted successfully',mtInformation,[mbOk],0);
  qryRec.Active := True;
end;                                                                    (4)
//=========================================================================
                                                           //Question 1.6
procedure TfrmRec.btnFClick(Sender: TObject);
begin
  qryRec.Active := False;
  qryRec.SQL.Text := 'SELECT Name,Surname,AgencyName,Province ✓FROM
tblAgencies, tblClients✓ WHERE tblAgencies.AgencyName✓ = tblClients.PlacedBy✓
AND (Province = 'Western Cape'✓ OR   Province = 'Gauteng') '✓;
  qryRec.Active := True;

end;                                                                    (6)
```

```
//============================================================================
procedure TfrmRec.btnGClick(Sender: TObject);                //Question 1.7
var
  sName, sDate :string;
begin
  sName := InputBox('Recruitment Agency', 'Enter the name of the agency', '');
✓
  sDate := InputBox('Recruitment Agency', 'Enter the cut off date', ''); ✓
  qryRec.Active := False;
  qryRec.SQL.Text :=   'SELECT Name, Surname, DatePlaced✓ FROM tblClients
✓WHERE DatePlaced < #' + sDate + '# ✓✓AND PlacedBy  ="'+ sName +'" ✓✓ ';
  qryRec.ExecSQL;
  qryRec.Active := True;
end;                                                                    (8)
//============================================================================
end.
```

TOTAL QUESTION 1: 35

QUESTION 2: OBJECT-ORIENTED PROGRAMMING (DELPHI)

unit uCityXXXX;

```
interface

uses SysUtils;
//============================================================================
// Q 2.1.1                          (3)
type TCity = class
    private ✓
      cityName : String;
      degreeJobs : integer;
      diplomaJobs : integer;        ✓✓
      salaryTotal : real;
    public
      constructor Create; overload;
      constructor Create(sCity : String); overload;
      procedure addDipJob(rSalary : real);
      procedure addDegJob(rSalary : real);
      function averageSalary : real;
      function getCityName : String;
      function isMatchCity(rSalary : real; sJobType : String) : boolean;
      function toString : String;
  end;

implementation

{ City }
//============================================================================
// Q 2.1.2                          (3)
constructor TCity.Create(sCity: String); ✓
begin
   cityName := sCity; ✓
   degreeJobs := 0;
   diplomaJobs := 0;                ✓
   salaryTotal := 0.00;
end;

constructor TCity.Create;
begin

end;
//============================================================================
// Q 2.1.3                          (5)
procedure TCity.addDegJob(rSalary: real);
begin
   inc(degreeJobs);
   salaryTotal := salaryTotal + rSalary;
end;                                          ✓ remove comment signs

procedure TCity.addDipJob(rSalary: real);
begin
   inc(diplomaJobs);
   salaryTotal := salaryTotal + rSalary;
end;

function TCity.averageSalary✓: real; ✓
begin
  Result := Round(salaryTotal / (diplomaJobs + degreeJobs)✓ * 100.0) / 100.0; ✓
end;
//============================================================================
```

```
// Q 2.1.4                          (2)
function TCompetitor.getCityName: String; ✓
begin
  Result := cityName ✓
end;
//============================================================================
// Q 2.1.5                          (5)
function TCity.isMatchCity(rSalary: real; sJobType: String✓): boolean✓;
begin
if ((UpperCase(sJobType) = 'DEGREE') AND (degreeJobs > diplomaJobs) AND
(averageSalary > rSalary))✓ then
    begin
      Result := true;
    end
  else
if ((UpperCase(sJobType) = 'DIPLOMA') AND (diplomaJobs > degreeJobs) AND
(averageSalary > rSalary))✓ then
    begin
      Result := true;
    end
  else
    Result := false; ✓
end;
//============================================================================
//Q 2.1.6                          (4)
function TCity.toString: String;
var
  objStr : String;
begin
  objStr := 'City : ' + cityName + #13; ✓
  objStr := objStr+'Diploma Jobs : '+intToStr(diplomaJobs)+ #9 +'Degree Jobs :
                              ' +  ✓intToStr(degreeJobs) + #13;
  objStr := objStr + 'Average Salary : R' ✓+ floatToStr(averageSalary); ✓

  Result := objStr;
end;
//============================================================================
```

unit Question2_UXXXX;

```
interface

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

type
  TfrmQuest2 = class(TForm)
    redOutput: TRichEdit;
    MainMenu1: TMainMenu;
    mnuA: TMenuItem;
    mnuB: TMenuItem;
    Quit: TMenuItem;
    function jobCategory(sJob : String) : String;
    procedure QuitClick(Sender: TObject);
    procedure DisplayCityInfoClick(Sender: TObject);
    procedure CheckJobMatchClick(Sender: TObject);
    procedure FormActivate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
var

  frmQuest2: TfrmQuest2;

implementation

{$R *.dfm}

uses
  uCityXXXX;
//============================================================================
```

Q 2.2.1 **(20)**

```
var
  city : TCity;                ✓✓
const
  degreeJobs : array[1..6] of String = ('Doctor', 'Programmer', 'Architect',
'Teacher', 'Lawyer', 'Engineer');
  diplomaJobs : array[1..6] of String = ('Secretary', 'Mechanic',
'Electrician', 'Beautician', 'Nurse', 'Plumber');

function TfrmQuest2.jobCategory(job: String): String;
var
  k : integer;
begin
   Result := 'Error';
   for k := 1 to 6 do
     begin
       if (sJob = degreeJobs[k]) then
         Result := 'Degree';
       if (sJob = diplomaJobs[k])  then
         Result := 'Diploma';
     end;
end;

procedure TfrmQuest2.QuitClick(Sender: TObject);
begin
  Application.Terminate;
end;

procedure TfrmQuest2.FormActivate(Sender: TObject);
var
  tFile : textfile; ✓
  sCity, sJob, sSalary, sCategory : String;
  iCounter : integer;
  rSalary :real;
begin
  redOutput.Lines.Clear;
  iCounter := 0; ✓
  if FileExists('Jobs.txt') <> TRUE then✓
    begin
      ShowMessage('File not found'); ✓
      Application.Terminate;
    end
  else
    begin
      AssignFile(tFile, 'Jobs.txt'); ✓
      Reset(tFile);

      ReadLn(tFile, sCity); ✓
      city := TCity.Create(sCity); ✓✓

      while NOT EOF(tFile) do ✓
```

```
        begin
          ReadLn(tFile, sJob); ✓
          ReadLn(tFile, sSalary); ✓


          rSalary = StrToFloat(sSalary); ✓
          sCategory := jobCategory(sJob); ✓

          if (sCategory = 'Degree') then
            begin
              city.addDegJob(rSalary);
            end
          else if (sCategory  = 'Diploma') then     ✓✓
            begin
              city.addDipJob(rSalary);
            end;
          inc(iCounter); ✓
        end;

      CloseFile(tFile); ✓
      redOutput.Lines.Add('');              ✓
      redOutput.Lines.Add(IntToStr(iCounter)+' jobs processed from '+
                                                          city.getCityName);
    end;
end;
//============================================================================
```

// Q 2.2.2 (2)

```
procedure TfrmQuest2.mnuAClick(Sender: TObject);
begin
   redOutput.Lines.Clear;
   redOutput.Lines.Add(city.toString); ✓✓
end;
//============================================================================
```

// Q 2.2.3 (5)

```
procedure TfrmQuest2.mnuBClick(Sender: TObject);
var
  sCategory : String;
  rSalary : real;
begin
  redOutput.Lines.Clear;

  sCategory := InputBox('Test City', 'Enter your qualification
                                        (Degree/Diploma)','');✓
  rSalary := StrToFloat(InputBox('Test City', 'Enter your minimum
                                          required salary', '')); ✓

  if (city.isMatchCity✓(rSalary, sCategory)✓) then
    begin
        redOutput.Lines.Add(city.getCityName + ' is a good place
                                          to look for a job');
    end
  else                                                                    ✓
    begin
      redOutput.Lines.Add(city.getCityName + ' does not
                                      meet your minimum requirements');
    end;
end;
end.
//============================================================================
```

TOTAL QUESTION 2: 49

QUESTION 3: DELPHI PROGRAMMING

NOTE: This is only a sample – learners may answer this question in any way they see fit. Allocate marks according to principles applied correctly. Can use the rubric supplied.

```
unit Question3_U;

interface

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

type
  TfrmQuestion3 = class(TForm)
    redOutput: TRichEdit;
    pnlButtons: TPanel;
    btnA: TButton;
    btnB: TButton;
    BitBtn1: TBitBtn;
    procedure FormCreate(Sender: TObject);
    procedure btnListClick(Sender: TObject);
    procedure btnNumbersClick(Sender: TObject);

    procedure FormActivate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmQuestion3: TfrmQuestion3;

implementation
//============================================================================
{$R *.dfm}
```

Q 3.1 (12)

```
type
  arrType = array[1..20] of string;
  Alpha = Set of 'A'..'Z';
var
  arrLearners : array[1..20] of string;
  arrVisitors :arrType; ✓
  iCount  :integer;
  Consonants : Alpha;
```

Note:
Transfer of parameter values to the Sort subprogram is not a requirement. The following declaration is acceptable:

```
arrVisitors: array[1..20] of string;
```

```
procedure TfrmQuestion3.FormCreate(Sender: TObject);
var
   iComma, iColon, iMark, K :integer;
   sSubject, sName:string;

 begin
  arrLearners[1] := 'Susan Thompson,Maths:77';
  arrLearners[2] := 'Eric Ntumba,IT:89';
  arrLearners[3] := 'Sean Franklin,Accounting:70';
  arrLearners[4] := 'Mohammed Naidoo,Maths:68';
  arrLearners[5] := 'Rowan Huntley,IT:77';
  arrLearners[6] := 'Elizabeth Xaba,Economics:77';
```

```
  arrLearners[7] := 'Sue Daniels,IT:69';
  arrLearners[8] := 'Zane Shameez,Maths:90';
  arrLearners[9] := 'Mpho Anderson,Science:81';
  arrLearners[10] := 'Bryan Smith,IT:75';
  arrLearners[11] := 'Christopher Khumalo,Accounting:78';
  arrLearners[12] := 'Camilla Johnson,Science:88';
  arrLearners[13] := 'Taryn Peterson,Science:70';
  arrLearners[14] := 'Jack Nelson,Maths:68';
  arrLearners[15] := 'Joe Zimmerman,Science:76';
  arrLearners[16] := 'Gregory Thompson,IT:87';
  arrLearners[17] := 'Dwane Franklin,IT:89';
  arrLearners[18] := 'Sean Franklin,Accounting:70';
  arrLearners[19] := 'Mohammed Naidoo,Maths:68';
  arrLearners[20] := 'Cindy Mokoena,IT:70';

   iCount := 0; ✓
   for K := 1 to 20 do✓
     begin
      iPlace := pos(',', arrLearners[k]); ✓
      sName := copy(arrLearners[K],1,iPlace - 1); ✓
      delete(arrLearners[K],1,iPlace);   } ✓
      iPlace := pos(':', arrLearners[K]);}
      sSubject := copy(arrLearners[K],1, iPlace -1); ✓
      iMark := StrToInt(copy(arrLearners[K], iPlace + 1, 3)); ✓

       if ((sSubject ='Maths') or (sSubject = 'Science') or (sSubject = 'IT'))
                                                  ✓ and (iMark >= 70) ✓ then
         begin
           inc(iCount); ✓
           arrVisitors[iCount] := sName; ✓
         end;
     end;
end;
```

Q 3.1: 12 marks
To test conditions and assign name to Visitors array
(1) Declare visitors array
(1) Initialize the counter
(1) Loop
Extract the following from string in given array
(2) name
(2) subject
(1) mark
(2) if testing conditions
    if true then
    (1) inc counter
    (1) assign name to visitors array

Delphi:
Solution without Delete-statement to extract name, subject and mark.
See comment block below.

Alternative: To extract the name, subject and mark do the following
Declare iComma and iColon as integer variables

```
iComma := pos(',', arrLearners[k]);
sName := copy(arrLearners[K],1,iComma - 1);
iColon := pos(':', arrLearners[K]);
sSubject := copy(arrLearners[K], iComma + 1, iColon – iComma - 1);
iMark := StrToInt(copy(arrLearners[K], iColon + 1, 3));
```

```
//============================================================================
```

Question 3.2 (6)

```
procedure Sort(var arrNames:arrType; iCount:integer);
var
      K, J  :integer;
      sTemp :string;

begin
  for K := 1 to iCount -1 do✓
    for J := k + 1 to iCount do✓
       if arrNames[K] > arrNames[J] then✓
          begin
            sTemp := arrNames[K]; ✓
            arrNames[K] := arrNames[J]; ✓
            arrNames[J] := sTemp; ✓
          end;
end;
```

Delphi:
Transfer of parameters is the best solution but not a requirement. Do not penalise for no parameters being used.

Q 3.2: 6 marks for subprogram
(1)Outer loop
(1)Inner loop
(1) if
Inside if:
(3) swap elements correctly

Can use any recognised sorting code.


```
//=============================================================================
// Question 3.3            (4)
procedure TfrmQuestion3.btnAClick(Sender: TObject);
var
    K :integer;


begin
     redOutput.Clear;
     Sort(arrVisitors, iCount); ✓
     redOutput.Lines.Add('Alphabetically Sorted List of Visitors'); ✓
     redOutput.Lines.Add('==================================');
      for K := 1 to iCount do✓
         redOutput.Lines.Add(arrVisitors[K]); ✓
end;
//=============================================================================
// Question 3.4            (14)
procedure TfrmQuestion3.btnBClick(Sender: TObject);
var
  K, J :integer;
  sNumber,sNum :string;

begin
  Consonants := ['B'..'D', 'F'..'H', 'J'..'N', 'P'..'T', 'V'..'Z'];
  Sort(arrVisitors, iCount); ✓
  redOutput.Clear;
  redOutput.Lines.Add('List of Visitors with Student Numbers');
  redOutput.Lines.Add('==================================');
  redOutput.Lines.Add('Name' + #9 + 'Student Number'); ✓
  redOutput.Lines.Add('==========================');

  for K := 1 to iCount do✓
    begin
      sNumber := '';
      for J := 1 to length(arrVisitors[K]) do ✓
      begin
       if (upcase(arrVisitors[K][J]) ✓ in Consonants✓) ✓
         then
          sNumber := sNumber  + upcase✓ (arrVisitors[K][J]);
      end;
      sNumber := copy(sNumber,1,3) ;
      sNumber := sNumber + ✓IntToStr(random(900) + 100); ✓✓

      redOutput.Lines.Add(arrVisitors[K] + #9✓ + sNumber); ✓
    end;
    redOutput.Lines.Add(' ');
  end;

//=============================================================================
```

Q 3.3: 4 marks for Display Names
(1)Sort array – call subprogram
(1)Display heading
(1) Loop
(1) Inside loop: display

Delphi: Alternative:
Can use a set with vowels instead of consonants and make use of NOT in the if-statement.

Q 3.4: 14 marks for Student Numbers
(1) Sort Array (call subprogram)
(2) Heading and subheadings
(1) Loop to iCount
Inside loop:
Get Consonants:
(1) Loop / repeat 3x
(3) get character,test for consonant & add to string
(1) capital letters
Get 3 digit number:
(2) generate random value
(1) join consonants and number together in single string
(2) neatly display name and number

Alternative for inner loop instead of FOR loop. Can use a WHILE loop:

```
 iCounter := 0;
 J := 1;
 while (iCounter < 3) and (J <=length(arrVisitors)) do
  begin
   if NOT(upcase(arrVisitors[K][J]) in ['A','E','I','O','U',' ']) then
     begin
      sNummer := sNummer + upcase(arrVisitors [K][J]);
      inc(iCounter);
     end;
   inc(J);
  end;
```

```
procedure TfrmQuestion3.FormActivate(Sender: TObject);
begin
  redOutput.Paragraph.TabCount := 1;
  redOutput.Paragraph.Tab[0] := 100;
end;

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

TOTAL QUESTION 3: 36

TOTAL SECTION A: 120

SECTION B: JAVA

QUESTION 1: PROGRAMMING AND DATABASE (JAVA)

```
   import java.io.*;
   import java.sql.*;
   import javax.swing.*;
   import java.util.Scanner;

    public class TestRecruitment
   {
       public static void main (String[] args) throws SQLException,IOException
      {
   BufferedReader inKb = new BufferedReader (new InputStreamReader
(System.in));
      Recruitment DB = new Recruitment();
      System.out.println();
      char choice = ' ';
        do
         {
            System.out.println("       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 = inKb.readLine().toUpperCase().charAt(0);
            System.out.println(" ");
            String sql = "";
            switch(choice)
            {
               case 'A':                                          //Question 1.1
                 {
         sql = "SELECT *✓FROM tblAgencies✓ORDER BY NumPrevPlacements✓
                                                                    DESC✓";
                    DB.A(sql);
                    break;
                 }                                                          (4)
//============================================================================
               case 'B':                                          //Question 1.2
                  {
              sql = "SELECT Name, Surname, Salary✓ FROM tblClients✓ WHERE
                    FullTime = False✓ AND Salary < 15000✓";

                    DB.B(sql);
                    break;
                 }                                                          (4)
//============================================================================
              case 'C':                                          //Question 1.3
                  {
              sql = "SELECT Count(*)✓ AS [Count] ✓FROM tblAgencies ✓
                    WHERE International = True"✓;
                    DB.C(sql);
                    break;
                 }                                                          (4)
```

```
//============================================================================
               case 'D':                                          //Question 1.4
                  {
             sql = "SELECT Name,Surname,Salary✓,round✓(salary * (10 / 100),2)✓
                       AS [AgentComm] ✓ FROM tblClients✓";
                                                    OR:(salary * 0.1, 2)
                     DB.D(sql);
                     break;
                  }                                                      (5)
//============================================================================
               case 'E':                                          //Question 1.5
                  {
                     Scanner kb = new Scanner (System.in);
                     System.out.println();

               sql = "INSERT INTO tblAgencies ✓
                     VALUES ✓('Jobs Unlimited','Western Cape',False,0) ✓✓";
                     DB.E(sql);

                     break;
                  }                                                      (4)
//============================================================================
               case 'F':                                          //Question 1.6
                  {
              sql = "Select Name,Surname,AgencyName,Province ✓from
                     tblAgencies, tblClients ✓where tblAgencies.AgencyName✓
                     = tblClients.PlacedBy ✓AND (Province = 'Western Cape'✓
                     OR Province = 'Gauteng') ✓";

                     DB.F(sql);

                     break;
                  }                                                      (6)
//============================================================================

               case 'G':         //Question 1.7
                  {
                     Scanner kb = new Scanner (System.in);
                     System.out.println("Enter the name of the agency ");
                     String agencyName = kb.nextLine();✓
                     System.out.println("Enter the date");
                     String date = kb.next();✓

                     sql = "SELECT Name,Surname,DatePlaced✓ FROM tblClients✓
                                 WHERE DatePlaced < #"+date+"#✓✓AND
                                    PlacedBy='"+agencyName+"'"✓✓;

                     DB.G(sql);
                     break
                  }                                                      (8)
//============================================================================
            }
         }while (choice != 'Q');

         DB.disconnect();
         System.out.println("Done");
      }
   }
//============================================================================
```

TOTAL QUESTION 1: 35

QUESTION 2: OBJECT-ORIENTED PROGRAMMING (JAVA)

City.java

```
//============================================================================
public class City
{
    // Q 2.1.1                                  (3)
    private✓ String cityName;
    private int degreeJobs;                     ✓✓
    private int diplomaJobs;
    private double salaryTotal;

    public City()
    {

    }
//============================================================================
    // Q 2.1.2                                  (3)
    public City(String name) ✓
    {
        cityName = name;✓
        degreeJobs = 0;
        diplomaJobs = 0;                        ✓
        salaryTotal = 0;
    }
//============================================================================
    // Q 2.1.3                                  (5)
    public void addDipJob(double salary)
    {
        diplomaJobs++;
        salaryTotal = salaryTotal + salary;
    }                                           ✓ Remove comment signs

    public void addDegJob(double salary)
    {
        degreeJobs++;
        salaryTotal = salaryTotal + salary;
    }

    public double✓ averageSalary()✓
    {
        double salary = Math.round(salaryTotal / (degreeJobs + diplomaJobs)* ✓
                                                              100) / 100.0;✓
        return salary;
    }
//============================================================================
    // Q 2.1.4                                  (2)
    public String getCityName()✓
    {
        return cityName; ✓
    }
 //============================================================================
    // Q 2.1.5                                  (5)
    public boolean✓ jobMatch(double sal, String type✓)
    {
        if (type.equalsIgnoreCase("degree") & degreeJobs > diplomaJobs &
                                                    averageSalary() > sal) ✓
        {
            return true;
        }
```

```
        else if (type.equalsIgnoreCase("diploma") & diplomaJobs > degreeJobs &
                                                        averageSalary() > sal) ✓
        {
            return true;
        }
        else
        {
            return false; ✓
        }
    }
 //===========================================================================
```

//Q 2.1.6 **(4)**

```
    public String toString()
    {
        String objStr = "";

        objStr = objStr + "City : " + cityName + "\n";✓
        objStr = objStr + "Diploma Jobs : " + diplomaJobs + "\t"  ✓
                  + "Degree Jobs : " + degreeJobs + "\n";
        objStr = objStr + "Average Salary : R"✓ + averageSalary() + "\n";✓

        return objStr;
    }
}
//===========================================================================
```

TestCity.java

```
import javax.swing.*;
import java.io.*;
import java.util.*;

public class TestCity
{
    public static String degreeJobs[] = {"Doctor", "Programmer", "Architect",
"Teacher", "Lawyer", "Engineer"};
    public static String diplomaJobs[] = {"Secretary", "Mechanic",
"Electrician", "Beautician", "Nurse", "Plumber"};

    public static String jobCategory(String jobType)
    {
        String jobType = "Error";

        for(int i = 0; i < 6; i++)
        {
            if (degreeJobs[i].equals(job))
            {
                jobType = "Degree";
            }
            else if (diplomaJobs[i].equals(job))
            {
                jobType = "Diploma";
            }
        }
        return jobType;
    }
```

```
//============================================================================
    // Q 2.2.1                          (20)
    public static void main(String args[]) throws Exception
    {
        City city = new City();✓✓

        File file = new File("Jobs.txt"); ✓

        if (!file.exists())✓
        {
            System.out.println("File not found"); ✓
            System.exit(0);
        }
        else
        {
            Scanner sc = new Scanner(file); ✓
            String sCity = sc.nextLine();✓

            city = new City(sCity); ✓✓

            int counter = 0; ✓

            while (sc.hasNextLine())✓
            {
                String job = sc.nextLine();✓
                double salary = Double.parseDouble(sc.nextLine());✓✓

                String jobType = jobCategory(job); ✓

                if (jobType.equals("Degree"))
                {
                    city.addDegJob(salary);
                }
                else if (jobType.equals("Diploma"))
                {
                    city.addDipJob(salary);
                }                                          ✓✓

                counter++; ✓
            }

            System.out.println(counter + " job opportunities processed from " +
                               city.getCityName());✓

            System.out.println();
            sc.close();          ✓
        }
        file.close();
//============================================================================
  BufferedReader inKb = new BufferedReader (new InputStreamReader (System.in));
  char ch = ' ';
        while (ch != 'Q')
        {

            System.out.println();
            System.out.println("       MENU");
            System.out.println(" ");
            System.out.println("       A – Option A");
            System.out.println("       B – Option B");
            System.out.println(" ");
            System.out.println("       Q - QUIT");
            System.out.println(" ");
```

```
            System.out.print("       Your Choice? :");

            ch = inKb.readLine().toUpperCase().charAt(0);

            switch (ch)
            {
//============================================================================
   // Q 2.2.2                          (2)
               case 'A':
               {
                   System.out.println();
                   System.out.println(city.toString());✓✓
                   break;
               }
  //============================================================================
   // Q 2.2.3                          (5)
             case 'B':
             {
              System.out.print("Enter your qualification(Degree/Diploma): ");
              String qual_type = inKb.readLine();✓
              System.out.print("Enter your minimum required salary : ");
              double money = Double.parseDouble(inKb.readLine());✓

              if (city.jobMatch✓(money, qual_type✓))
              {
                 System.out.println(city.getCityName() + " is a good place
                                          to look for a job");
              }
              else                                                          ✓
              {
                 System.out.println(city.getCityName() + " does not meet
                                          your minimum requirements");
              }
              break;
             }

                case 'Q':
                {
                    System.exit(0);
                } // case

            } // switch

        } // while

    } // main

} // class
```

TOTAL QUESTION 2: 49

QUESTION 3: JAVA PROGRAMMING

NOTE: This is only a sample – learners may answer this question in any way they see fit. Allocate marks according to principles applied correctly. Can use the rubric supplied.

TestVisitors.java

```
import java.util.Scanner;
import java.io.*;
import javax.swing.*;
//============================================================================
// Question 3.1          (12)
public class TestVisitors
{
   static String[] arrLearners = new String[20];
   static String[] arrVisitors = new String[20]; ✓

   public static void main(String[] args) throws Exception
   {
   arrLearners[0] =  "Susan Thompson,Maths:77";
   arrLearners[1] =  "Eric Ntumba,IT:89";
   arrLearners[2] =  "Sean Franklin,Accounting:70";
   arrLearners[3] =  "Mohammed Naidoo,Maths:68";
   arrLearners[4] =  "Rowan Huntley,IT:77";
   arrLearners[5] =  "Elizabeth Xaba,Economics:77";
   arrLearners[6] =  "Sue Daniels,IT:69";
   arrLearners[7] =  "Zane Shameez,Maths:90";
   arrLearners[8] =  "Mpho Anderson,Science:81";
   arrLearners[9] =  "Bryan Smith,IT:75";
   arrLearners[10]=  "Christopher Khumalo,Accounting:78";
   arrLearners[11] =  "Camilla Johnson,Science:88";
   arrLearners[12] =  "Taryn Peterson,Science:70";
   arrLearners[13] =  "Jack Nelson,Maths:68";
   arrLearners[14] =  "Joe Zimmerman,Science:76";
   arrLearners[15] =  "Gregory Thompson,IT:87";
   arrLearners[16] =  "Dwane Franklin,IT:89";
   arrLearners[17] =  "Sean Franklin,Accounting:70";
   arrLearners[18] =  "Mohammed Naidoo,Maths:68";
   arrLearners[19] =  "Cindy Mokoena,IT:70";

   int count = 0; ✓
   for (int k = 0; k < 20; k++)✓
    {
      int comma = arrLearners[k].indexOf(','); ✓
      String name = arrLearners[k].substring(0,comma); ✓

      int colon = arrLearners[k].indexOf(':'); ✓
      String subject = arrLearners[k].substring(comma + 1,colon ); ✓

      int mark = Integer.parseInt(arrLearners[k].substring(colon+1)); ✓
      if(((subject.equals("Maths")) || (subject.equals("Science")) ||
                (subject.equals("IT"))) ✓&& (mark >= 70)) ✓
       {
         arrVisitors[count] = name; ✓
         count++;✓
       }
    }
```

Q 3.1: 12 marks
(1) Declare Visitors array
(1) Initialize the counter
(1) Loop
Extract the following from string in given array
(2) name
(2) subject
(1) mark
(2) if testing conditions
    if true then
    (1) inc counter
    (1) assign name to visitors array

```
Scanner input = new Scanner(System.in);
char ch = ' ';
while (ch != 'Q')
 {
    System.out.println();
    System.out.println("           MENU");
    System.out.println(" ");
    System.out.println("       A – Option A");
    System.out.println("       B – Option B");
    System.out.println(" ");
    System.out.println("       Q - QUIT");
    System.out.println(" ");
    System.out.print("       Your Choice? :");

    ch = input.nextLine().toUpperCase().charAt(0);
    switch (ch)
    {
     case 'A':
     {
//=========================================================================
// Question 3.3          (14)
       sort(arrVisitors, count); ✓
       System.out.println(" ");
       System.out.println("Sorted List of Visitors");✓
       System.out.println("======================");
       for (int  k = 0;k < count; k++)✓
        {
         System.out.println(arrVisitors[k]); ✓
        }
        break;
      }
//=========================================================================
// Question 3.4
      case 'B':
      {
        sort(arrVisitors, count); ✓
        System.out.println(" ");
        System.out.println("List of Visitors");✓
        System.out.println("================");
        System.out.printf("%-25s%-10s\n","Name",
                                    "Student Number"); ✓
        System.out.println("============================
                                           ==========");
        String number;
        for (int k = 0;k < count; k++)✓
         {
           number = "";

           for (int j = 0; j < arrVisitors[k].length();
                                          j++)✓//OR while
           {
             char letter = arrVisitors[k].toUpperCase()✓
                                             .charAt(j); ✓
             if (!(letter =='A')&& !(letter=='E')&&
                  !(letter=='I')&& !(letter=='O')&&
                    !(letter=='U')&& !(letter==' ')) ✓
                  number = number + letter; ✓
```

Q 3.3: 4 marks for Display Names
(1)Sort array
(1)Heading
(1) Loop
(1) Inside loop: display

Q 3.4: 14 marks for Student Numbers
(1) Sort Array
(2) Heading and subheadings
(1) Loop to iCount
Inside loop:
Get Consonants:
(1) Loop/repeat 3x
(3) get character,test for consonant & add to string
(1) capital letters
Get 3-digit number:
(2) generate random value
(1) join consonants and number
(2) neatly display name and number

```
            }
            number = number.substring(0,3) ;
            number = number + ✓ (int) (Math.random () * (900) + 100); ✓✓

            System.out.printf("%-25s%-10s\n"✓,arrVisitors[k],number✓);
          }
          break;
       }

       case 'Q':
       {
         System.exit(0);
       } // case

      } // switch

  } // while
 }


//=========================================================================
// Question 3.2          (6)

public static void sort(String [] arrNames, int count)
{
  for (int k = 0; k < count -1; k++)✓
    for (int j = k + 1; j < count; j++)✓
      {
         if (arrNames[k].compareTo(arrNames[j]) > 0) ✓
           {
           String temp = arrNames[k]; ✓
           arrNames[k] = arrNames[j]; ✓
           arrNames[j] = temp; ✓
           } // end if
      }  // end for
  } // end method

}

//=========================================================================
```

Q 3.2: 6 marks for sorting method
(1)Outer loop
(1)Inner loop
(1) if
Inside if:
(3) elements correctly swapped

Can use any recognised sorting method.

TOTAL QUESTION 3: 36

TOTAL SECTION B: 120

ANNEXURE A

QUESTION 1: DELPHI – PROGRAMMING AND DATABASE

CENTRE NUMBER: ……	EXAMINATION NUMBER: ……

QUESTION 1: DELPHI – MARKING GRID

QUESTION	ASPECT	MAX. MARKS	LEARNER'S MARKS
1.1	'SELECT *✓ FROM tblAgencies✓ ORDER BY✓ NumPrevPlacements DESC'✓;	**4**	
1.2	'SELECT Name, Surname, Salary✓ FROM tblClients✓ WHERE FullTime = False✓ AND Salary < 15000✓'	**4**	
1.3	'SELECT Count(*)✓ AS [The number of agencies that offer international jobs are] ✓ FROM tblAgencies✓ WHERE International=True✓'	**4**	
1.4	'SELECT Name,Surname,Salary✓,format✓ (salary * (10 / 100),".00"✓)AS [AgentComm] ✓ FROM tblClients✓' OR (salary * 0.1 , 2)	**5**	
1.5	'INSERT INTO tblAgencies✓ VALUES✓ ("Jobs Unlimited","Western Cape",False,0) ✓✓ '	**4**	
1.6	'SELECT Name,Surname,AgencyName,Province ✓FROM tblAgencies, tblClients ✓WHERE tblAgencies.AgencyName✓ = tblClients.PlacedBy✓ AND (Province = "Western Cape" ✓OR Province = "Gauteng"✓) ';'	**6**	
1.7	Input name, ✓ input date✓ 'SELECT Name,Surname,DatePlaced✓ FROM tblClients ✓WHERE DatePlaced < #' + sDate + '# ✓✓AND PlacedBy ="'+ sName +'" ✓✓ '	**8**	
	TOTAL:	**35**	

ANNEXURE B

QUESTION 2: DELPHI – OBJECT-ORIENTED PROGRAMMING

CENTRE NUMBER:……	EXAMINATION NUMBER: ……

QUESTION 2: DELPHI – MARKING GRID

QUESTION	ASPECT	MAX. MARKS	LEARNER'S MARKS
2.1			
2.1.1	**Define attributes for TCity**: four private (1) fields, with the correct type (1) and names (1)	**3**	
2.1.2	**Constructor:** name parameter (1) assigns value of parameter to cityName field (1) initialises other fields to zero (1)	**3**	
2.1.3	**averageSalary function:** uncommented addDipJob and addDegJob functions (1). Method called averageSalary (1) which returns real (1) containing correct calculation (1) and rounded to two decimal places (1)	**5**	
2.1.4	**getCityName function:** Correct name and returns String (1), Correct field returned (1)	**2**	
2.1.5	**jobMatch function:** receives one real and one String parameter (1) and returns boolean value (1) correct conditions for determining matches (2) and returns true/false correctly (1)	**5**	
2.1.6	**toString function:** adds tabs (1) new lines (1) to existing code and appends average salary (1) with heading and tab (1)	**4**	
2.2			
2.2.1	• Declare a single TCity object (2) • Create a new variable for file (1) • Create an counter integer and initialises it to zero (1) • If file does not exist (1) display message and exit (1) • If file exists then assignfile (1) • Read out first line outside of loop into city name variable (1) • Call constructor Create of TCity (1) using name as a parameter (1) • Loop (1) Read line from file for job (1) and salary (1) which is converted to a real (1), • Call jobCategory (1) and compare result (1) call appropriate method (addDegJob/addDipJob) in TCity (1), • Increase counter (1) and display result of processing (1) • Close file outside loop (1)	**20**	
2.2.2	Call toString method of TCity object (2)	**2**	
2.2.3	• Read qualification (1) and salary (1) from the keyboard • Call the isMatchCity method in the TCity class (1) with two correct parameters (1) • Display correct output based on returned Boolean value (1)	**5**	
	TOTAL:	**49**	

ANNEXURE C

QUESTION 3: DELPHI PROGRAMMING

CENTRE NUMBER: ……..	EXAMINATION NUMBER: ……..		
QUESTION 3: DELPHI – MARKING GRID			
QUESTION	**ASPECT**	**MAX. MARKS**	**LEARNER'S MARKS**
3.1	**12 marks to evaluate learners and place names in Visitors array** (1) Declare visitors array (1) Initialize the counter (1) Loop to read through learners array Get the following from the string in the given array (2) name (2) subject (1) mark (2) if testing conditions if true then (1) inc counter (1) assign name to visitors array	**12**	
3.2	**6 marks for subprogram** (1) Outer loop (1) Inner loop (1) if Inside if: (3) elements correctly swapped	**6**	
3.3	**4 marks for Display Names** (1) Sort array (call subprogram) (1) Heading (1) Loop (1) Inside loop: display	**4**	
3.4	**14 marks for Student Numbers** (1) Sort Array (call subprogram) (2) Heading and subheadings (1) Loop to iCount Inside loop: Get Consonants: (1) Loop/repeat 3x (3) Test for consonant (1) capital letters Get 3 digit number: (2) generate random value (1) join consonants and number to form single string (2) display name and number neatly	**14**	
	TOTAL:	**36**	

ANNEXURE D

QUESTION 1: JAVA – PROGRAMMING AND DATABASE

CENTRE NUMBER: ……	EXAMINATION NUMBER:……

QUESTION 1: JAVA – MARKING GRID

QUESTION	ASPECT	MAX. MARKS	LEARNER'S MARKS
1.1	SELECT * ✓ FROM tblAgencies✓ ORDER BY NumPrevPlacements ✓DESC✓	**4**	
1.2	SELECT Name, Surname, Salary✓ FROM tblClients ✓ WHERE FullTime=False✓ AND Salary < 15000✓	**4**	
1.3	SELECT Count(*)✓ AS [Count] ✓ FROM tblAgencies✓ WHERE International = True✓	**4**	
1.4	SELECT Name,Surname,Salary✓, round✓ (salary * (10 / 100),2) ✓AS [AgentComm] ✓ FROM tblClients✓ **OR** (salary * 0.1 , 2)	**5**	
1.5	INSERT INTO tblAgencies ✓ VALUES ✓ ('Jobs Unlimited','Western Cape',False,0) ✓✓	**4**	
1.6	SELECT Name,Surname,AgencyName,Province ✓FROM tblAgencies, tblClients✓ WHERE tblAgencies.AgencyName✓ = tblClients.PlacedBy✓ AND (Province = 'Western Cape'✓ OR Province = 'Gauteng')	**6**	
1.7	Input Agencyname ✓ Input Date ✓ SELECT Name,Surname,DatePlaced✓ FROM tblClients✓ WHERE DatePlaced < #"+date+"#✓✓ AND PlacedBy = '"+agencyName+"'"✓✓	**8**	
	TOTAL:	**35**	

ANNEXURE E

QUESTION 2: JAVA – OBJECT-ORIENTED PROGRAMMING

CENTRE NUMBER:……	EXAMINATION NUMBER: ……

QUESTION 2: JAVA – MARKING GRID

QUESTION	ASPECT	MAX. MARKS	LEARNER'S MARKS
2.1			
2.1.1	**Define attributes for City**: four private (1) fields, with the correct type (1) and names (1)	**3**	
2.1.2	**Constructor:** name parameter (1) assigns value of parameter to cityName field (1) initialises other fields to zero (1)	**3**	
2.1.3	**averageSalary method:** uncommented addDipJob and addDegJob methods (1). Method called averageSalary (1) which returns double (1) containing correct calculation (1) and rounded to two decimal places (1)	**5**	
2.1.4	**getCityName method:** Correct name and returns String (1), Correct field returned (1)	**2**	
2.1.5	**jobMatch method:** receives one double and one String parameter (1) and returns boolean (1) correct conditions for determining matches (2) and returns true/false correctly (1)	**5**	
2.1.6	**toString method:** adds tabs (1) new lines (1) to existing code and appends average salary (1) with heading and tab (1)	**4**	
2.2			
2.2.1	• Declare a single City object (2) • Creating a new File (1) • Create an counter integer and initialises it to zero (1) • If file does not exist (1) display message and exit (1) • If file exists then create new Scanner (1) • Read out first line outside of loop into city name variable (1) • Call constructor City (1) using name as a parameter (1) • Loop to read from file (1) • Read line from file for job (1) and salary (1) which is converted to a double (1), • Call jobCategory (1) and compare result (1) call appropriate method (addDegJob/addDipJob) in City (1), • Increase counter (1) and display result of processing (1) • Close file outside loop (1)	**20**	
2.2.2	Call toString method of City object (2)	**2**	
2.2.3	• Read qualification (1) and salary (1) from keyboard • Call the isMatchCity method in the City class (1) with two correct parameters (1) • Display the correct output based on returned Boolean (1)	**5**	
	TOTAL:	**49**	

ANNEXURE F

QUESTION 3: JAVA PROGRAMMING

CENTRE NUMBER: ……..	EXAMINATION NUMBER: ……..

QUESTION 3: JAVA – MARKING GRID

QUESTION	ASPECT	MAX. MARKS	LEARNER'S MARKS
3.1	**12 marks to evaluate learners and place names in Visitors array** (1) Declare visitors array (1) Initialize the counter (1) Loop to read through learners array Get the following from the string in the given array (2) name (2) subject (1) mark (2) if testing conditions if true then (1) inc counter (1) assign name to visitors array	**12**	
3.2	**6 marks for subprogram** (1) Outer loop (1) Inner loop (1) if Inside if: (3) elements correctly swapped	**6**	
3.3	**4 marks for Display Names** (1) Sort array (call method) (1) Heading (1) Loop (1) Inside loop: display	**4**	
3.4	**14 marks for Student Numbers** (1) Sort Array (call method) (2) Heading and subheadings (1) Loop to iCount Inside loop: Get Consonants: (1) Loop/repeat 3x (3) Test for consonant (1) capital letters Get 3 digit number: (2) generate random value (1) join consonants and number to form single string (2) display name and number neatly	**14**	
	TOTAL:	**36**	