education

Department:
Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

FEBRUARY/MARCH 2010

MEMORANDUM

MARKS: **120**

The memorandum consists of 25 pages and 7 annexures.

General information:

- **Pages 2 – 11 contain the Delphi memoranda of possible solutions for QUESTIONS 1 to 3 in programming code.**
- **Pages 12 – 25 contain the Java memoranda of possible solutions for QUESTIONS 1 to 3 in programming code.**
- **Pages 26 – 32 contain Annexures A to G which includes a cover sheet as well as 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 uFrmMain;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, DB, ADODB, Grids, DBGrids, ExtCtrls, Buttons;
type
  TfrmFF = class(TForm)
    Panel1: TPanel;
    grdFF: TDBGrid;
    Panel2: TPanel;
    btnAll: TButton;
    btnDiff: TButton;
    tblFF: TDataSource;
    btnDelete: TButton;
    btnLow: TButton;
    btnHistory: TButton;
    BitBtn1: TBitBtn;
    btnAdjust: TButton;
    qryFF: TADOQuery;
    procedure btnAllClick(Sender: TObject);
    procedure btnDiffClick(Sender: TObject);
    procedure btnDeleteClick(Sender: TObject);
    procedure btnLowClick(Sender: TObject);
    procedure btnHistoryClick(Sender: TObject);
    procedure btnAdjustClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  frmFF: TfrmFF;
implementation

{$R *.dfm}
procedure TfrmFF.btnAllClick(Sender: TObject);    //Question 1.1
begin
  qryFF.Active := False;
  qryFF.SQL.Text := 'SELECT ✓*✓  FROM tblPeople✓ ORDER BY✓ EntryDate DESC✓';
  qryFF.Active := True;
end;                                                                    (5)
//=====================================================================
```

```
procedure TfrmFF.btnDiffClick(Sender: TObject);   //Question 1.2
begin
  qryFF.Active := False;
  qryFF.SQL.Text := 'SELECT PersonID, Name, S_Weight, G_Weight,  ✓, ' +
      'ROUND✓ (S_Weight - G_Weight✓, 1)  ✓  AS KgsToLose ✓FROM tblPeople✓';
 qryFF.Open;
end;                                                                              (6)
//=========================================================================
procedure TfrFFL.btnDeleteClick(Sender: TObject);  //Question 1.3
begin
  qryFF.Active := False;
  qryFF.SQL.Text := 'DELETE ✓FROM tblPeople ✓WHERE✓ Balance > 400✓ and ✓Name <>
"Uriel Knight"'; ✓
  qryFF.ExecSQL;
  ShowMessage('Overdue accounts deleted.');

 end;                                                                             (6)
//=========================================================================
procedure TfrmFF.btnLowClick(Sender: TObject);  //Question 1.4
begin
  qryFF.Active := False;
  qryFF.SQL.Text := 'SELECT MIN(Weight)✓ AS MinWeight✓,PersonID✓  FROM ' +
                    'tblWeighin✓ GROUP BY✓ PersonID✓';
  qryFF.ExecSQL;
  qryFF.Open;
end;                                                                              (6)
//=========================================================================
procedure TfrmFF.btnHistoryClick(Sender: TObject);  //Question 1.5
begin
var
  id : string;
begin
  id := InputBox('Individual weigh-in history ','Enter the ID ', '1');  ✓
  qryFF.Active := False;
  qryFF.SQL.Text := 'SELECT tblPeople.PersonID, ✓Name, G_Weight, Weight, ' +
                    'WeighDate ✓FROM tblPeople,  ✓ tblWeighIn✓ WHERE ' +
                    'tblPeople.PersonID✓ = tblWeighin.PersonID✓  AND✓ ' +
                    'tblPeople.PersonID ✓= '+ id;  ✓

  qryFF.ExecSQL;
  qryFF.Open;
end;                                                                             (10)
//=========================================================================
procedure TfrmFF.btnAdjustClick(Sender: TObject);  //Question 1.6
begin
  qryFF.Active := False;
  qryFF.SQL.Text := 'UPDATE tblWeighin ✓SET✓ Weight✓ = Weight ✓* 1.1✓ WHERE ' +
                    'MONTH✓   (WeighDate) = 5✓';
  qryFF.ExecSQL;
end;                                                                              (7)
//=========================================================================

end.
```

QUESTION 2: OBJECT-ORIENTED PROGRAMMING (DELPHI)

unit MealXXXX;

```
interface
uses
  SysUtils;
Type                                          // Q 2.1.1.        (6 ÷ 2 = 3)
   TMeal = class✓ (TObject)
     private✓
      fDay         :String;  }     ✓
      fMealTime    :String;  }
      fFats        :integer; }
      fCarbs       :integer; }     ✓
      fProtein     :integer; }
     public✓
      constructor Create (sDay, sMeal : String;iFat,iProt,iCarbs :
                                                  integer) ✓;
      function getDay : String;
      function noFats : boolean;
      function calculatePoints : integer;
      function toString:String;
  end;

implementation
```

// Q 2.1.2. **(4 ÷ 2 = 2)**

```
constructor TMeal.Create✓ (aDay, sMeal : String;iFat,iProt,iCarbs :
                                                  integer) ✓;
begin
  fDay := sDay;
  fMealTime := sMeal;
  fFats := iFat;                    ✓✓
  fProtein := iProt;
  fCarbs := iCarbs;
end;
```

// Q 2.1.3. **(4 ÷ 2 = 2)**

```
function TMeal.getDay✓: String; ✓
begin
  result✓ := fDay; ✓
end;
```

// Q 2.1.4. **(4 ÷ 2 = 2)**

```
function TMeal.noFats: boolean✓;
begin
  if fFats = 0✓ then
       result := true✓
  else
      result := false✓;
end;
```

OR

```
 result := false✓;
 if fFats = 0✓ then
    result := true;✓
```

// Q 2.1.5. (6 ÷ 2 = 3)

```
function TMeal.calculatePoints: integer✓;
  var tot : integer;
begin
  tot := (fFats * 4) + (fProtein * 2) + (fCarbs * 2); ✓✓
  if noFats then
       tot := tot - 2; ✓
  if fFats > 2 then
      tot := tot + 10; ✓
  result := tot; ✓
end;
```

// Q 2.1.6. (6 ÷ 2 = 3)

```
function TMeal.toString:String✓;
begin
  result := fDay✓ + #9✓ + intToStr✓ (fFats) + #9 + intToStr(fProtein) +
  #9 + intToStr(fCarbs) + #9 + intToStr (calculatePoints) ✓ // call
  method
  + #9 + fMealTime; ✓ // for all the fields
end;
end.
```

```
//=====================================================================
unit QuestTwoXXXX_U;

interface

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

type
  TfrmQuest2 = class(TForm)
    MainMenu1: TMainMenu;
    DailyReport: TMenuItem;
    NoFats: TMenuItem;
    Quit1: TMenuItem;
    redOutput: TRichEdit;
    BestWorst: TMenuItem;
    procedure Quit1Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure DailyReportClick(Sender: TObject);
    procedure NoFatsClickSender: TObject);
    procedure BestWorstClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmQuest2: TfrmQuest2;

implementation
//=====================================================================
```

// Q 2.2.1. (26 ÷ 2 = 13)

```
uses MealXXXX; ✓
var
  arrMeals :array[1..100] of TMeal✓;
  iCount    :integer✓;

{$R *.dfm}

procedure TfrmQuest2.FormActivate(Sender: TObject);
var
  TFile :TextFile✓;
  sLine, sDay, sMealTime :String;
  iFats, iProtein, iCarbs, iHash  :integer;

begin
  if fileExists('Meals.txt') ✓ <> true then
    begin
      ShowMessage('The text file ''Meals.txt'' does not exist'); ✓
      Exit; ✓
    end;
    AssignFile(TFile, 'Meals.txt') ✓;
    Reset(TFile) ✓;
    iCount := 0; ✓
    While not eof(TFile) ✓ do
      begin
        inc(iCount); ✓// MUST initialise before loop and increment inside loop
        readln(TFile, sLine) ✓;
        iHash := pos ('#', sLine); ✓
        sDay := copy(sLine, 1, iHash -1); ✓
        delete(sLine, 1, iHash); ✓

        iHash := pos ('#', sLine);
        sMealTime := copy(sLine, 1, iHash -1); ✓
        delete(sLine, 1, iHash); ✓

        iHash := pos ('#', sLine);
        iFats := StrToInt✓ (copy(sLine, 1, iHash -1)); ✓
        delete(sLine, 1, iHash);

        iHash := pos ('#', sLine);
        iProtein := StrToInt(copy(sLine, 1, iHash -1)); ✓
        delete(sLine, 1, iHash);

        iCarbs := StrToInt(sLine); ✓

        arrMeals[iCount] ✓ := TMeal.Create✓ (sDay, sMealTime, iFats,
                                              iProtein, iCarbs) ✓;
     end;
     CloseFile(TFile); ✓
end;
```

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

// Q 2.2.2. (14 ÷ 2 = 7)

```
procedure TfrmQuest2.DailyReportClick OptionA1Click(Sender: TObject);
var
  K          : integer;
  iTotal     : integer;
  sDayOfWeek : String;
begin
  sDayOfWeek := InputBox('Day', 'Enter the Day (e.g. Mon, Tue, etc)', 'Sun'); ✓
  iTotal := 0; ✓
  redOutput.Clear;
  redOutput.Lines.add('Information for Meals for ' + uppercase(sDayOfWeek)); ✓
  redOutput.Lines.add('');
  redOutput.Lines.add('Day'+ #9 + 'Fats' + #9 +'Prot'+ #9 +
                      'Carbs' + #9 + 'Points' + #9 + 'Meal'); ✓
  For K := 1 to iCount do✓
  begin
    if arrMeals[K].getDay✓ = sDayOfWeek✓ then
      begin
        redOutput.Lines.add(arrMeals[K].toString) ✓;
        iTotal := iTotal✓ + arrMeals[K].calculatePoints✓;
      end;
    end;
     redOutput.Lines.Add(' ');
    if iTotal = 0 then
      ShowMessage('No Meals found for ' + sDayOfWeek)
    else
     redOutput.Lines.add('The total number of points is ' + IntToStr(iTotal))✓;
    if iTotal > 50 then ✓
      redOutput.Lines.add('Limit Exceeded') ✓
    else
      redOutput.Lines.add('Within Limit'); ✓
end;
```

OR

```
if iTotal <= 50 then
     redOutput.Lines.add('Within
Limit')
else
 redOutput.Lines.add('Limit
Exceeded');
```

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

// Q 2.2.3. (6 ÷ 2 = 3)

```
procedure TfrmQuest2.NoFatsClick(Sender: TObject);
var
  K  :integer;
begin
  redOutput.Clear;
  redOutput.Lines.add('Information on Meals with No Fats.'); ✓
  redOutput.Lines.add('');
  redOutput.Lines.add('Day'+ #9 + 'Fats' + #9 +'Prot'+ #9 + ✓
                      'Carbs' + #9 + 'Points' + 'Meal');
  For K := 1 to iCount do✓
  begin
    if arrMeals[K] ✓.NoFats✓ then
      begin
        redOutput.Lines.add(arrMeals[K].toString); ✓
      end;
    end;
end;
```

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

// Q 2.2.4. (12 ÷ 2 = 6)

```
procedure TfrmQuest2.BestWorstClick(Sender: TObject);
var
    K, iHighest, iLowest : integer;
    sHighest, sLowest    : String;
begin
  redOutput.Clear;
  redOutput.Lines.add('Meals with the Most and Least Points:');
  redOutput.Lines.add('');
  redOutput.Lines.add(#9 + 'Day'+ #9 + 'Fats' + #9 +'Prot'+ #9 +
                      'Carbs' + #9 + 'Points' + 'Meal');
  redOutput.Lines.add('');
  iHighest := arrMeals[1].calculatePoints;
  iLowest  := arrMeals[1].calculatePoints;
  sHighest := arrMeals[1].ToString;
  sLowest  := arrMeals[1].ToString;
  For K := 2 to iCount do✓
  begin
    if arrMeals[K].calculatePoints✓ > iHighest✓ then
      begin✓
        iHighest := arrMeals[K].calculatePoints; ✓
        sHighest := arrMeals[K].ToString; ✓
      end;
    if arrMeals[K].calculatePoints < iLowest then
      begin
        iLowest := arrMeals[K].calculatePoints;
        sLowest := arrMeals[K].ToString;

      end;
    end;
    redOutput.Lines.add('Highest' + #9 + sHighest) ✓;
    redOutput.Lines.add('Lowest' + #9 + sLowest); ✓
end;
procedure TfrmQuest2.Quit1Click(Sender: TObject);
begin
  Close;
end;

end.
```

✓

✓✓ **Initialise variables before loop**

The begin must be with the end in correct position for 1 mark

✓ **Repeat with < for Lowest**

Do not penalise here again for previous errors

QUESTION 3: DELPHI PROGRAMMING

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

```
unit SMSCompXXXX_U;

interface

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

type
  TfrmSMSComp = class(TForm)
    MainMenu1: TMainMenu;
    Processing: TMenuItem;
    ExtractPossibleWinners: TMenuItem;
    GetAndDisplayWinners: TMenuItem;
    redDisplay: TRichEdit;
    SaveWinners: TMenuItem;
    Quit1: TMenuItem;

    procedure ExtractPossibleWinnersClick(Sender: TObject);
    procedure GetAndDisplayWinnersClick(Sender: TObject);
    procedure CellNumbersofpossiblewinners1Click(Sender: TObject);
    procedure SaveWinnersClick(Sender: TObject);
    procedure Quit1Click(Sender: TObject); private
    procedure FormCreate(Sender: TObject); private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmSMSComp: TfrmSMSComp;
  arrEntries : array[1..20] of string;
  arrCorrect : array[1..20] of string; ✓
  arrWinners : array[1..3] of string;
  iCounter, iCount :integer;


implementation

{$R *.dfm}

procedure TfrmSMSComp.FormCreate(Sender: TObject);
begin
      arrEntries[1] := '082 345 4571:Exercise';
      arrEntries[2] := '082543 2341:Exercise';
      arrEntries[3] := '082 234 1241:EXERCISE';
      arrEntries[4] := '0821239876:Eat';
      arrEntries[5] := '083123 6123:Sleep';
      arrEntries[6] := '083 452 4353:EXERCISE';
      arrEntries[7] := '0831009844:Sleep';
      arrEntries[8] := '083 104 2333:Exercise';
      arrEntries[9] := '076 239 6966:Sleep';
      arrEntries[10] := '076986 4533:EAT';
      arrEntries[11] := '076 365 4272:Exercise';
      arrEntries[12] := '076563 2642:Exercise';
      arrEntries[13] := '084 884 1244:EXERCISE';
```

```
      arrEntries[14] := '0841239867:Sleep';
      arrEntries[15] := '084123 6444:Exercise';
      arrEntries[16] := '084 1156 434:Exercise';
      arrEntries[17] := '079 1229 844:Eat';
      arrEntries[18] := '079 456 2331:Exercise';
      arrEntries[19] := '079 239 7971:EXERCISE';
      arrEntries[20] := '079986 6622:EAT';
      iCounter := 20;
end;
//============================================================================
function removeSpaces(sNumber:string):string; ✓
var
  K         :integer;
  sNewNumber:string;
begin
  sNewNumber := ''; ✓
  for K := 1 to length(sNumber) do✓
    begin
      if (sNumber[K] <> ' ') ✓ then
       sNewNumber := sNewNumber + sNumber[K];✓
    end;
    Result := sNewNumber; ✓
end;
//============================================================================
procedure TfrmSMSComp.ExtractPossibleWinnersClick(Sender: TObject);
var
  K, iPosColon   :integer;
  sNumber, sAnswer:string;

begin
 iCount := 0; ✓
 for K := 1 to iCounter do✓
   begin
    iPosColon := pos(':', arrEntries[K]); ✓
    sNumber := copy(arrEntries[K], 1,iPosColon - 1);✓
    sNumber := removeSpaces(sNumber); ✓
    delete(arrEntries[K], 1, iPosColon); ✓
    sAnswer := arrEntries[K]; ✓

    if (uppercase✓ (sAnswer)  = 'EXERCISE') ✓ then
      begin      // must initialize iCount
        iCount := iCount + 1; ✓
        arrCorrect[iCount] := sNumber; ✓
      end;
    end;



 redDisplay.Clear;
  redDisplay.Lines.Add('Cellphone numbers of possible
                                          winners');
  for K := 1 to iCount do                              ✓✓
     redDisplay.Lines.Add(arrCorrect[K]);

end;
//============================================================================
```

Q 3.1 : 6 marks for subprogram
(1)Function definition
(1)Initialize new string
(1) Loop through number
Inside loop:
(1) check if character not space then
(1) add character to new string
After loop:
(1) return new string

Q 3.1 : 13 marks
(1) Loop through given array
(2) extract cellnumber
(1) Call function to remove spaces
(2) extract answer
(1) if answer is correct
    (1) ignoring case

Avoid duplication (3 marks):
EITHER
Declare new array(1)
use counter for new array correctly(1)
insert into new array (1)
OR
Testing for duplicates in existing array(1) and deleting duplicates from existing array(2)

Display info: heading and loop(1), display(1)

```
procedure TfrmSMSComp.SelectAndDisplayWinnersClick(Sender: TObject);
var
   K, iRandomNumber  :integer;
begin
  redDisplay.Lines.Add(' ');
  redDisplay.Lines.Add('List of winners'); ✓

  Randomize;
  for K := 1 to 3 do✓
   begin
      repeat
        iRandomNumber := Random(iCount) + 1; ✓
      until arrCorrect[iRandomNumber] <> ''; ✓




      if (arrCorrect[iRandomNumber] <> '') then✓
        begin


           arrWinners[K]:= arrCorrect[iRandomNumber]; ✓
           redDisplay.Lines.Add('Winner # ' + IntToStr(K) + ' : '✓ +
                                                 arrWinners[K]); ✓
         arrCorrect[iRandomNumber] ✓ := ''; ✓
        end;
       end;
     end;

//============================================================================
procedure TfrmSMSComp.SaveWinnersClick(Sender: TObject);

 var
  fOut  :TextFile;
  K :integer;

begin
  AssignFile(fOut, 'Winners.txt'); ✓
  Rewrite(fOut); ✓
  writeln(fOut, 'List of winners'); ✓
  for K := 1 to 3 do✓
    begin
      writeln(fOut, ✓ 'Winner # ' + IntToStr(K) + ' : ' + arrWinners[K]); ✓
    end;
    CloseFile(fOut); ✓
end;
//============================================================================
procedure TfrmSMSComp.Quit1Click(Sender: TObject);
begin
  Close;
end;

end.
```

Q 3.2 : 10 marks
Use new array / 3 variables
(1)Loop 3 times / assign to 3 variables
(2) Generate random number

(4) Any way of excluding winner to be selected again
POSSIBLE SOLUTION:
Use repeat until to test if number chosen is chosen already e.g. not blank in old array (1)
If not found, place selected number in new array(1) and clear number / remove number(1) from old array(1)

(1) display heading
(1) display winners
(1) in correct format

Q 3.3: 7 marks
(1) use correct filename
(1) open file for output
(1) write heading to file
(2) write to file 3 times
(1) correct format
(1) close file

TOTAL SECTION A: 120

SECTION B: JAVA

QUESTION 1: PROGRAMING AND DATABASE (JAVA)

```
import java.io.*;
import java.sql.*;

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

     FatFighters DB = new FatFighters();
     System.out.println("\f");

     char choice = ' ';
     do
     {
        System.out.println("       MENU");
        System.out.println();
         System.out.println("    A - All Members");
         System.out.println("    B - Weight Differences");
         System.out.println("    C - Overdue Accounts");
         System.out.println("    D - Lowest Weight");
         System.out.println("    E – Weigh-in History");
        System.out.println("    F - Adjust Weights");
        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 tblPeople✓ ORDER BY ✓EntryDate
                                                          DESC✓";
                 DB.all(sql);
                break;
            }                                                    (5)

      //============================================================
            case 'B':                                    //Question 1.2
            {
                 sql = "SELECT PersonID, Name, S_Weight, G_Weight✓,
                       ROUND✓(S_Weight - G_Weight✓, 1)✓ AS KgsToLose✓
                       FROM tblPeople✓";

                 DB.weightDiff(sql);
                break;
            }                                                    (6)

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

```
            case 'C':                                   //Question 1.3
            {
                  sql = " DELETE✓ FROM tblPeople✓ WHERE✓ Balance > 400✓
                              AND✓ Name <> 'Uriel Knight'✓";

                  DB.overdue(sql);
                 break;
            }                                                       (6)

      //============================================================
            case 'D':                                   //Question 1.4
            {
                  sql = " SELECT MIN(Weight) ✓ AS MinWeight✓, PersonID✓
                              FROM tblWeighIn✓ GROUP BY✓ PersonID✓
                        “;

                  DB.lowest(sql);
                 break;
             }                                                      (6)

      //============================================================
            case 'E':                                   //Question 1.5
            {
                  System.out.print("Enter the PersonID of the member: ");
                  String id = inKb.readLine();✓
                  sql = " SELECT tblPeople.PersonID✓, Name, G_Weight,
                        WeighDate✓, Weight FROM tblPeople✓, tblWeighin✓
                        WHERE tblPeople.PersonID✓ = tblWeighin.PersonID✓
                        AND✓ tblPeople.PersonID✓ = " + id✓;

                  DB.history(sql);
                 break;                                         (10)

      //============================================================
            }
            case 'F':                                   //Question 1.6
            {
                  sql = "UPDATE tblWeighin✓ SET✓ Weight✓ = Weight✓ *
                              1.1✓ WHERE MONTH✓ (WeighDate)=5✓";

                  DB.adjust(sql);

                  break;
             }                                                      (7)

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

            }

       }
        while (choice != 'Q');

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

```
 import java.sql.*;
 import java.io.*;
 import javax.swing.JOptionPane;

   public class FatFighters
   {
      Connection conn;


      public FatFighters ()
      {
      //load the driver
         try
         {
            Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
            System.out.println ("Driver successfully loaded");
         }
         catch (ClassNotFoundException c)
         {
           System.out.println ("Unable to load database driver");
         }

            //connect to the database
     try
     {

      //conn = DriverManager.getConnection
("jdbc:odbc:FatFighters.mdb");

     //System.out.print("Type in the exact location of your database
(FOR
                                  EXAMPLE - C:/TEST/FatFighters.mdb)");
     //BufferedReader inKb = new BufferedReader (new InputStreamReader
                                                  (System.in));
     //String filename = inKb.readLine();

       String filename = "FatFightersDB.mdb";

       String database = "jdbc:odbc:Driver={Microsoft Access Driver
           (*.mdb)};DBQ=";
       database += filename.trim () + ";DriverID=22;READONLY=true}";
       conn = DriverManager.getConnection (database, "", "");

       System.out.println ("Connection to database successfully
                                             established");

    }
    catch (Exception e)
   {
         System.out.println ("Unable to connect to the database");
   }

 } //end connect

 public void all (String sql) throws SQLException
      {
         System.out.println();
         Statement stmt = conn.createStatement ();
```

```
         ResultSet rs = stmt.executeQuery (sql);
         System.out.printf("%-10s%-18s%-10s%-10s%-12s%-
10s","PersonID","Name","S_Weight", "G_Weight","EntryDate", "Balance");
         System.out.println();

System.out.println("==========================================================
==================");

         while (rs.next ())
         {
            String sId = rs.getString ("PersonID");
            String sName = rs.getString ("Name");
            String sInitial = rs.getString ("S_Weight");
            String sGoal = rs.getString ("G_Weight");
            String sEntryDate = rs.getString("EntryDate");
            sEntryDate = sEntryDate.substring(0,10);
            double dBal = Double.parseDouble(rs.getString("Balance"));
            dBal = Math.round(dBal * 100) / 100.0;
            System.out.printf("%-10s%-18s%-10s%-10s%-12s %-10s",sId
,sName,sInitial, sGoal,sEntryDate, dBal);
            System.out.println();
         }

         System.out.println(" ");
         rs.close();
         stmt.close ();
      }

      public void weightDiff (String sql)throws SQLException
      {
         Statement stmt = conn.createStatement ();

         ResultSet rs = stmt.executeQuery (sql);
         System.out.printf("%-10s%-18s%-10s%-10s%-
10s","PersonID","Name","S_Weight", "G_Weight", "KgsToLose");
         System.out.println();

System.out.println("==========================================================
=========");
         while (rs.next ())
         {
            String sId = rs.getString ("PersonID");
            String sName = rs.getString ("Name");
            String sInitial = rs.getString ("S_Weight");
            String sGoal = rs.getString ("G_Weight");
            String sKgs = rs.getString ("KgsToLose");

          //int kgs = (int)Double.parseDouble(sKgs);


            System.out.printf("%-10s%-18s%-10s%-10s%-10s",sId
,sName,sInitial, sGoal, sKgs);
            System.out.println();
         }
         System.out.println(" ");
         rs.close();
         stmt.close ();
      }
```

```
      public void overdue (String sql)throws SQLException
      {
         System.out.println();
         Statement stmt = conn.createStatement ();

         int rows = stmt.executeUpdate (sql);

         System.out.println(rows + " records deleted");

         stmt.close ();
      }



      public void lowest(String sql)throws SQLException
      {
         System.out.println();
         Statement stmt = conn.createStatement ();

         ResultSet rs = stmt.executeQuery (sql);
         System.out.printf("%-10s%-14s","PersonID","MinWeight");
         System.out.println();
         System.out.printf("========================");
         System.out.println();
         while (rs.next ())
         {
            String sID = rs.getString ("PersonID");
            double dWeight =
Double.parseDouble(rs.getString("MinWeight"));
            int Weight = (int)Math.round(dWeight);

            System.out.printf("%-10s%-14s",sID, Weight);
                System.out.println();
         }
         System.out.println(" ");
         stmt.close ();
      }

       public void history(String sql)throws SQLException
      {
         System.out.println();
         Statement stmt = conn.createStatement ();

         ResultSet rs = stmt.executeQuery (sql);
         System.out.printf("%-10s%-18s%-10s%-10s%-
14s","PersonID","Name", "G_Weight", "Weight","WeighDate");
         System.out.println();

System.out.println("==================================================
========");
         while (rs.next ())
         {

            String sId = rs.getString ("PersonID");
            String sName = rs.getString ("Name");
            String sGoal = rs.getString ("G_Weight");
            double dWeight = Double.parseDouble(rs.getString("Weight"));
            int Weight = (int)Math.round(dWeight);
```

```
            String sWeighDate = rs.getString("WeighDate");
            sWeighDate = sWeighDate.substring(0,10);

            System.out.printf("%-10s%-18s%-10s%-10s%-14s",sId,sName,
sGoal, Weight,sWeighDate);
            System.out.println();
         }
         System.out.println(" ");
         stmt.close ();
      }
      public void adjust(String sql)throws SQLException
      {
         System.out.println();
         Statement stmt = conn.createStatement ();

         int rows = stmt.executeUpdate (sql);

         System.out.println(rows + " records updated");

         sql = "SELECT * FROM tblWeighin";

         ResultSet rs = stmt.executeQuery (sql);
         System.out.printf("%-10s%-15s%-14s","PersonID","WeighDate",
"Weight");
         System.out.println();
         System.out.printf("==============================");
         System.out.println();
         while (rs.next ())
         {
            String sID = rs.getString ("PersonID");
            String sWeighDate = rs.getString("WeighDate");
            sWeighDate = sWeighDate.substring(0,10);
            String sWeight = rs.getString ("Weight");

            System.out.printf("%-10s%-15s%-14s",sID, sWeighDate,
sWeight);
                System.out.println();
         }
         System.out.println(" ");
         stmt.close ();
      }

    public void disconnect () throws SQLException
    {

        conn.close ();
    }

   }
```

QUESTION 2: OBJECT-ORIENTED PROGRAMMING

MealXXXX.java

// Q 2.1.1. (6 ÷ 2 = 3)

```
public class✓ MealXXXX
{
      private✓ String day; ✓
      private String meal;
      private int fats; ✓
      private int proteins;
      private int carbs;
```

✓ **Class name same as file name**
✓ **Methods public**

// Q 2.1.2. (4 ÷ 2 = 2)

```
      public MealXXXX(✓String day, String mtime, int fat, int prot, int carb)
✓
      {
            day = day;
            meal = mtime;
            fats = fat;
            proteins = prot;
            carbs = carb;
      }
```

✓✓

// Q 2.1.3. (4 ÷ 2 = 2)

```
      public String✓ getDay()✓
      {
            return✓ day;✓
      }
```

// Q 2.1.4. (4 ÷ 2 = 2)

```
      public boolean✓ noFats()
      {
            if (fats == 0) ✓
            {
                  return true✓;
            }
            else
            {
                  return false✓;
            }
      }
```

OR

```
boolean result = false✓;
 if (fFats == 0)✓
    result = true;✓
 return result;
```

// Q 2.1.5. (6 ÷ 2 = 3)

```
      public int✓ calculatePoints()
      {
            int points = fats * 4 + proteins * 2 + carbs * 2✓✓;

            if (fats > 2)
            {
                  return (points + 10); ✓
            }
            else if (noFats())
            {
                  return (points - 2); ✓
            }
            else
            {
                  return points; ✓
            }
      }
```

// Q 2.1.6. (6 ÷ 2 = 3)

```
      public String✓ toString()
      {
            return day✓ + "\t\t"✓ + fats ✓+ "\t\t" + proteins + "\t\t\t" +
                  carbs + "\t\t" + calculatePoints()✓// call method
                        + "\t\t\t" + meal; ✓ // all the fields
      }
} // end of class

//-------------------------------------------------------------------------
-
```

TestMeal.java

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

public class TestMeals
{
      public static void main(String args[]) throws Exception
      {
```

// Q 2.2.1. (26 ÷ 2 = 13)

```
            MealXXXX[]✓ arrMeal = new MealXXXX[100]; ✓
            int counter = 0; ✓

            File f✓ = new File("Meals.txt");✓

            if (f.exists())✓ // with else for 1 mark
            {
               Scanner sc = new Scanner(f); ✓
               while ✓ (sc.hasNextLine())✓
                  {
                        String line = sc.nextLine();✓

                        String tokens[]✓= line.split✓ ("#");✓

                        String day = tokens[0]; ✓
```

```
                  String meal = tokens[1]; ✓
                  int fats = Integer.parseInt✓ (tokens[2]); ✓
                  int proteins = Integer.parseInt(tokens[3]); ✓
                  int carbs = Integer.parseInt (tokens[4]); ✓

                  arrMeal[counter✓] = new✓ MealXXXX(day, meal, fats,
                  proteins, carbs); ✓
                  counter++;✓
            }

            sc.close(); ✓
      }
      else
      {
            System.out.println("File does not exist");✓
            System.exit(0);   ✓
      }

      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 - Daily Report");
            System.out.println("       B - Meals Without Fats");
            System.out.println("       C - Best and Worst Meals");
            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. (14 ÷ 2 = 7)

```
            case 'A':
            {
                  System.out.println();

                  System.out.print("Enter a day : " );
                  String day = inKb.readLine().toUpperCase();✓
                  int totalPoints = 0; ✓

                  System.out.println();                                ✓
                  System.out.println("Information on Meals for " + day);
                  System.out.println();

                  System.out.println("Day\t\tFat\t\t
                              Protein\t\tCarbs\tPoints\t\tMeal");✓

                  System.out.println();


                  for(int i = 0; i < counter; i++)✓
```

```
                    {
                    if(arrMeal[i] ✓.getDay()✓.toUpperCase().equals(day) ✓)
                      {
                          totalPoints = totalPoints ✓ +
                                      arrMeal[i].calculatePoints()✓;

                          System.out.println(arrMeal[i].toString()) ✓;
                       }
                    }

                    System.out.println();
                    System.out.println("Daily Total : " + totalPoints); ✓

                    if (totalPoints > 50) ✓
                     {
                          System.out.println("Limit Exceeded");
                     }
                    else
                     {                                                    ✓
                          System.out.println("Within Limit");
                     }

                    break;
               }
```

OR

```
if totalPoints <= 50
{
System.out.println("With
in Limit");
}
else
{System.out.println("Lim
it Exceeded");
}
```

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

// Q 2.2.3. **(6 ÷ 2 = 3)**

```
                    case 'B':
                    {
                     System.out.println();
                     System.out.println("Information on Meals with No
                                                          Fats");✓
                     System.out.println();

     System.out.println("Day\t\tFat\t\tProtein\t\tCarbs\tPoints\t\tMeal");✓
                          System.out.println();

                          for(int i = 0; i < counter; i++)✓
                          {
                                if (arrMeal[i] ✓.noFats())✓
                                {
                          System.out.println(arrMeal[i].toString());✓
                                }
                          }

                          System.out.println();
                          break;
                    }
```

```
//=====================================================================
// Q 2.2.4.                          (12 ÷ 2 = 6)

                  case 'C':
                  {
                   System.out.println();
                   System.out.println("Meals with the Most and Least Points");
                   System.out.println();

      System.out.println("\t\tDay\t\tFat\t\tProtein\t\tCarbs\tPoints\t\tMeal");
                        System.out.println();

                        int maxPoints = arrMeal[0];
                        int maxDay = 0;
            // or String maxDay = arrMeal[0].toString();

                        int minPoints = arrMeal[0];
                        int minDay = 0;
            // or String minDay = arrMeal[0].toString();

                        for(int i = 1; i < counter; i++)
                         {
                           int points = arrMeal[i].calculatePoints();

                           if (points < minPoints)
                           {
                              minPoints = points;
                              minDay = i;
                        // or minDay = arrMeal[i].toString();
                              }

                              if (points > maxPoints)
                              {
                                    maxPoints = points;
                                    maxDay = i;
                        // or minDay = arrMeal[i].toString();

                              }
                        }

                  System.out.print("Highest\t");
                  System.out.println(mealArr[maxDay].toString());
                        // OR System.out.println(maxDay.toString());

                  System.out.print("Lowest\t");
                  System.out.println(mealArr[minDay].toString());
                        // OR System.out.println(minDay.toString());
                  System.out.println();
                  break;
            }
            case 'Q':
             {
                  System.exit(0);
             } // case
      } // switch
    } // while

  } // main

} // class
```

✓

✓✓ **Initialise variables before loop**

✓ **Repeat with < for Lowest**

Do not penalise here again for previous errors

//This bracket
//with the closing
//bracket for 1 mark

QUESTION 3: JAVA PROGRAMMING

TestSMSCompetitionXXXX.java

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

public class TestSMSCompetitionXXXX
{
   static String[] arrEntries = new String[20];
   static String[] arrCorrect = new String[20];
   static String[] arrWinners = new String[3];
   static int counter = 0;
   static String answer = "EXERCISE";

   public static void main(String[] args) throws Exception
   {

      arrEntries[0] = "082 345 4571:Exercise";
      arrEntries[1] = "082543 2341:Exercise";
      arrEntries[2] = "082 234 1241:EXERCISE";
      arrEntries[3] = "0821239876:Eat";
      arrEntries[4] = "083123 6123:Sleep";
      arrEntries[5] = "083 452 4353:EXERCISE";
      arrEntries[6] = "0831009844:Sleep";
      arrEntries[7] = "083 104 2333:Exercise";
      arrEntries[8] = "076 239 6966:Sleep";
      arrEntries[9] = "076986 4533:EAT";
      arrEntries[10] = "076 365 4272:Exercise";
      arrEntries[11] = "076563 2642:Exercise";
      arrEntries[12] = "084 884 1244:EXERCISE";
      arrEntries[13] = "0841239867:Sleep";
      arrEntries[14] = "084123 6444:Exercise";
      arrEntries[15] = "084 1156 434:Exercise";
      arrEntries[16] = "079 1229 844:Eat";
      arrEntries[17] = "079 456 2331:Exercise";
      arrEntries[18] = "079 239 7971:EXERCISE";
      arrEntries[19] = "079986 6622:EAT";


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 - Extract Possible Winners");
        System.out.println("       B - Select and Display Winners");
        System.out.println("       C - Save Winners");
        System.out.println(" ");
        System.out.println("       Q - QUIT");
        System.out.println(" ");
        System.out.print("       Your choice? :");

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

      switch (ch)
      {
         case 'A':
            {
```

```
            extractCorrect();
           break;
          }

        case 'B':
          {
            selectWinners();
            break;
          }

        case 'C':
          {
            writeWinners();
            break;
          }

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

     } // switch

    } // while
  }

 public static String removeSpaces(String s) ✓
  {
    String withoutSpaces = "";✓

    for(int i = 0; i < s.length(); i++)✓
    {

      if (s.charAt(i) != ' ') ✓
      {

          withoutSpaces = withoutSpaces + s.charAt(i); ✓
       }
     }

    return withoutSpaces; ✓
  }
//=========================================================
  public static void extractCorrect()
  {
    for(int i = 0; i < arrEntries.length; i++)✓
    {
      String pn = arrEntries[i].substring
                 (0,arrEntries[i].indexOf(":"));✓✓
      pn = removeSpaces(pn); ✓

      String ans = arrEntries[i].substring
                 (arrEntries[i].indexOf(":") + 1); ✓✓

      if (ans.equalsIgnoreCase✓ (answer)) ✓
       {
                arrCorrect[counter] = pn; ✓
                counter++;✓
       }
    }
```

Q 3.1 : 6 marks for method
(1)Definition of the method
(1)Initialize new string
(1) Loop through number
Inside loop:
(1) check if character not space then
(1) add character to new string
After loop:
(1) return new string

Q 3.1 : 13 marks
(1) Loop through given array
(2) Extract cellnumber
(1) Call function to remove spaces
(2) Extract answer
(1) if answer is correct
(1) ignoring case

Avoid duplication (3 marks):
EITHER
Declare new array(1)
use counter for new array correctly(1)
insert into new array (1)
OR
Testing for duplicates in existing array(1) and deleting duplicates from existing array(2)

Display info: heading and loop(1), display(1)

```
      System.out.println("Cellphone numbers of
                                  possible winners");

      for(int i = 0; i < counter; i++)✓
      {
            System.out.println(arrCorrect[i]); ✓
      }
   }
//============================================================================
   public static void selectWinners() throws Exception
   {
      for(int i = 0; i < arrWinners.length; i++)✓
      {
         int randPos = (int) (Math.random()✓ *
                                      counter); ✓
         arrWinners[i] = arrCorrect[randPos]; ✓


         for(int j = randPos; j < counter - 1; j++)✓
          {
            arrCorrect[j] = arrCorrect[j+1]; ✓
          }

      counter--;✓
      }
      System.out.println("\n");
      System.out.println("List of winners");✓

      for(int i = 0; i < arrWinners.length; i++)✓
      {
        System.out.println("Winner #" + (i+1) + " : " + arrWinners[i]); ✓
      }
   }
   //============================================================================
   public static void writeWinners() throws Exception
   {

            PrintWriter✓ fout = new PrintWriter(new File("Winners.txt"));✓
            fout.println("List of Winners");  ✓
            for(int i = 0; i < arrWinners.length; i++)✓
            {
              fout.println✓ ("Winner #" + (i+1) + " : " + arrWinners[i]); ✓

            }

            fout.close();✓
   }
}//============================================================================
=
```

Q 3.2 : 10 marks
Can create new array / use 3 variables
(1)Loop 3 times / assign to 3 variables
(2) Generate random number

(4) Any way of excluding winner to be selected again
POSSIBLE SOLUTION:
Use loop to test if number chosen is in old array (1)
If not, place selected number in new array(1) and remove number(1) from old array(1)

(1) display heading
(1) display winners
(1) in correct format

Q 3.3: 7 marks
(1) Use correct filename
(1) open file for output
(1) write heading to file
(2) write winners to file – repetition 3 times
(1) correct format
(1) close file

END OF SECTION B: JAVA **TOTAL SECTION B: 120**

ANNEXURE A

GRADE 12 MARCH 2010

INFORMATION TECHNOLOGY P1

COVER SHEET

Province: _____

Centre Number: _____

Examination Number: _____

Programming Language (circle the language used): DELPHI / JAVA

TOTAL MARKS PER QUESTION		
QUESTION	**MARK OUT OF**	**LEARNER'S MARK**
1	**40**	
2	**44**	
3	**36**	
GRAND TOTAL	**120**	

ANNEXURE B – March 2010

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 tblPeople✓ ORDER BY ✓EntryDate DESC✓	**5**	
1.2	SELECT PersonID, Name, S_Weight, G_Weight✓, ROUND✓(S_Weight - G_Weight✓, 1)✓ AS KgsToLose✓ FROM tblPeople✓	**6**	
1.3	DELETE✓ FROM tblPeople✓ WHERE✓ Balance > 400✓ AND✓ Name <> "Uriel Knight"✓;	**6**	
1.4	SELECT MIN(Weight) ✓ AS MinWeight✓, PersonID✓ FROM tblWeighIn✓ GROUP BY✓ PersonID✓	**6**	
1.5	Keyboard input of id from user ✓ SELECT tblPeople.PersonID✓, Name, G_Weight, WeighDate✓, Weight FROM tblPeople✓, tblWeighIn✓ WHERE tblPeople.PersonID✓ = tblWeighIn.PersonID✓ AND✓ tblPeople.PersonID✓ = ' + id✓	**10**	
1.6	UPDATE tblWeighIn SET✓ Weight✓ = Weight✓ * 1.1 ✓ WHERE MONTH✓✓ (WeighDate)=5✓	**7**	
	TOTAL:	**40**	

ANNEXURE C – March 2010

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 class (1) MealXXXX , five private (1) fields , 2 strings(1), 3 integers(1), Methods public (1) Constructor (1) **(6 ÷ 2 = 3)**	**3**	
2.1.2	**Constructor:** Parameters correct order (1), correct types(1) Assignment of fields (2) **(4 ÷ 2 = 2)**	**2**	
2.1.3	**getDay** (1) method, correct return type (1) Correct field (1) returned (1) **(4 ÷ 2 = 2)**	**2**	
2.1.4	**noFats method:** returns boolean(1) check if fats is zero(1) return true(1) else return false(1) **(4 ÷ 2 = 2)**	**2**	
2.1.5	**Calculatepoint:** returns an integer (1), correct base calculation (2) add 10 points for more than 2 fats (1) deduct 2 points for no fats (1) no change (1) **(6 ÷ 2 = 3)**	**3**	
2.1.6	**toString:** Returns String (1), with all fields (2), covert fields to integers where required(1), call calculatePoints method (1), tabs (1) **(6 ÷ 2 = 3)**	**3**	
2.2			
2.2.1	Uses class(1) Declare array of objects(1), Declare counter(1), Declare text file(1) if file not exists(1), display message (1), Exit(1) Before loop: Assignfile (1), Reset(1) Initialise counter(1), loop (1), Inside loop: Inc count(1) Read from file (1), Split string using pos of #(1), extract day with copy(1), delete(1), extract meal(2), extract fats(1), convert to int(1), extract proteins (1) and carbs(1), create object (1) add at correct position in the array(1) with arguments(1) Close file outside loop(1) **(26 ÷ 2 = 13)**	**13**	
2.2.2	Input Day (1), initialise total(1), Display headings(1), for loop(2) test array item using getDay(1) and compare it to user input(1) Inside if: Display with toString(1), call calculatePoints(1) and add to total (1), Outside for, display total(1) if totalpoints > 50(1) message(1) else other message(1) **(14 ÷ 2 = 7)**	**7**	
2.2.3	Print headings(2) Loop(1) test element of array(1) and call noFats method(1) test using if(1) and display inside if (1) **(6 ÷ 2 = 3)**	**3**	
2.2.4	Heading(1),Initialise two variables to store max and min(2), loop(1), call calculatePoints method(1) and test if points are less than min(1) Inside if (1) update minimum points (1) and update minDay(1) repeat the if-statement with > for maxpoints and maxday(1) Outside loop display max(1) and min(1) **(12 ÷ 2 = 6)**	**6**	
	TOTAL:	**44**	

ANNEXURE D – March 2010

QUESTION 3: DELPHI PROGRAMMING

CENTRE NUMBER: ……..	EXAMINATION NUMBER: …….		
QUESTION 3 DELPHI – MARKING GRID			
QUESTION	**ASPECT**	**MAX. MARKS**	**LEARNER'S MARKS**
3.1	**Subprogram: 6 marks** (1) Define subprogram (1) Initialize variable (1) Loop through number (1) Check if character not space then (1) Add character to new variable (1) Return variable **Extract Possible Winners : 13 marks** (1) Loop through given array (2) Extract number (1) Call function to remove spaces (2) Extract answer (1) Check if answer is correct (1) ignoring case Avoid duplication: 3 marks (1) Declare new array (1) use counter for new array correctly (1) insert into a new array OR use given array Display info: heading, loop(1), display(1)	**19**	
3.2	**Selects and Display Winners: 10 marks** (2) Generate random number (1) loop 3 times / other repetition such as 3 variables Any way of excluding winner to be selected again: 4 marks. One possible solution: Can create new array(1), Initialise and use new counter(1), Test(1), place in new array(1) Display heading(1), winners (1) in correct format(1)	**10**	
3.3	**Save Winners: 7 marks** (1) Use correct filename (1) Open file for output (1) Write heading to file (1) Write to file 3 times(1) correct format(1) (1) Close file	**7**	
	TOTAL:	**36**	

ANNEXURE E – March 2010

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 tblPeople✓ ORDER BY ✓EntryDate DESC✓	**5**	
1.2	SELECT PersonID, Name, S_Weight, G_Weight✓, ROUND✓(S_Weight - G_Weight✓, 1)✓ AS KgsToLose✓ FROM tblPeople✓	**6**	
1.3	DELETE✓ FROM tblPeople✓ WHERE✓ Balance > 400✓ AND✓ Name <> 'Uriel Knight'✓	**6**	
1.4	SELECT MIN(Weight) ✓ AS MinWeight✓, PersonID✓ FROM tblWeighin✓ GROUP BY✓ PersonID✓	**6**	
1.5	Keyboard input of id from user ✓ SELECT tblPeople.PersonID✓, Name, G_Weight, WeighDate✓, Weight FROM tblPeople✓, tblWeighin✓ WHERE tblPeople.PersonID✓ = tblWeighin.PersonID✓ AND✓ tblPeople.PersonID✓ = " + id✓	**10**	
1.6	UPDATE tblWeighin ✓SET✓ Weight✓ = Weight✓ * 1.1 ✓ WHERE MONTH✓ (WeighDate)=5✓	**7**	
	TOTAL:	**40**	

ANNEXURE F – March 2010

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 class (1) MealXXXX , five private (1) fields , 2 strings(1), 3 integers(1), Methods public (1), Class and file name the same(1) **(6 ÷ 2 = 3)**	**3**	
2.1.2	**Constructor:** Parameters correct order (1), correct types(1) Assignment of fields (2) **(4 ÷ 2 = 2)**	**2**	
2.1.3	**getDay** (1) Public (1) Correct return type (1) Correct field (1), returned (1) **(4 ÷ 2 = 2)**	**2**	
2.1.4	**noFats methods:** returns boolean(1) check if fats is zero(1) return true(1) else return false(1) **(4 ÷ 2 = 2)**	**2**	
2.1.5	**CalculatePoints:** returns an int (1), correct base calculation (2) add 10 points for more than 2 fats (1) deduct 2 points for no fats (1) no change (1) **(6 ÷ 2 = 3)**	**3**	
2.1.6	**toString:** Returns String (1), with attributes (2), calculated points (2) and tabs (1) **(6 ÷ 2 = 3)**	**3**	
2.2			
2.2.1	Declare array of objects(2) Initialise counter(1), Create File object(1) with correct file name(1) if file exists(1) then create Scanner /FileReader object(1) while loop (1) Read from file (1), Create split array(2) with # as delimiter(1), extract day(1), extract meal(1), extract fats(1) and convert to int(1), extract proteins(1), extract carbs(1) create object (1) at correct position in the array(1) with arguments(1) Inc counter(1), Close file outside loop(1), If file does not exist (1)display message(1) and exit(1) **(26 ÷ 2 = 13)**	**13**	
2.2.2	Inputs: Day (1), initialise total(1), headings(1), for loop(1) using counter (1), test array item i(1) using getDay(1) and compare it to user input(1) Inside if:call calculatePoints(1) and add to total (1), display with toString(1). Outside for, display total(1) if totalpoints > 50(1) message else other message(1) **(14 ÷ 2 = 7)**	**7**	
2.2.3	Print headings(2) Loop(1) test element of array using if(1) call noFats method(1) and display element of array(1) **(6 ÷ 2 = 3)**	**3**	
2.2.4	Display heading(1)Initialise two variable to store max and min(1) and max and min day(1) loop(1), call calculatePoints method(1), test if points are less than min(1) Inside if(1)update minimum(1) and related day(1) Repeat the if-statement with > to test max points (1) Outside loop display max(1) and min(1) from array. **(12 ÷ 2 = 6)**	**6**	
	TOTAL:	**44**	

ANNEXURE G – March 2010

QUESTION 3: JAVA PROGRAMMING

CENTRE NUMBER: ……..	EXAMINATION NUMBER: ……..		
QUESTION 3 JAVA – MARKING GRID			
QUESTION	**ASPECT**	**MAX. MARKS**	**LEARNER'S MARKS**
3.1	**Method: 6 marks** (1) Define method (1) Initialize variable (1) Loop through number (1) Check if character not space then (1) Add character to new variable (1) Return variable **Extract Possible Winners: 13 marks** (1) Loop through given array (2) Extract number (1) Call function to remove spaces (2) Extract answer (1) Check if answer is correct (1) ignoring case Avoid duplication: 3 marks (1) Declare new array (1) use counter for new array correctly (1) insert into a new array OR use given array Display info: heading, loop(1), display(1)	**19**	
3.2	**Selects and Display Winners: 10 marks** (2) Generate random number (1) loop 3 times / other repetition such as 3 variables Any way of excluding winner to be selected again: 4 marks. One possible solution: Can create new array(1), Initialise and use new counter(1), Test(1), place in new array(1) Display heading(1), winners (1) in correct format(1)	**10**	
3.3	**Save Winners: 7 marks** (1) Use correct filename (1) Open file for output (1) Write heading to file (1) Write to file 3 times(1)Correct format(1) (1) Close file	**7**	
	TOTAL:	**[36]**	