education

Department:
Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY

PAPER 1

MEMORANDUM

MARKS: 120

This memorandum consists of 30 pages.

SECTION A: DELPHI PROGRAMMING

QUESTION ONE: DELPHI DATABASE CONNECTIVITY

Mark Allocation

Question One - Marking Grid			
Question	**Aspect**	**Max Marks**	**Learner's Marks**
1.1	SQL: Select all fields(1) and display all the fields (1) sorted according to name of condition(1)	**3**	
1.2	SQL: Selected fields (1) from correct table(1) Age > 50 (1), "Lung%" (1)	**4**	
1.3	SQL: Update (1), SET field (1), condition (1)	**3**	
1.4	SQL: Insert (1), table VALUES (1), correct fields(1)	**3**	
1.5	Input(1) SQL: SELECT fields(1) from correct table(1) WHERE CdtnType = " ' + con + ' " (1) AND MONTH(DateAdmitted)(1) = " ' +monthNum +'" (1)	**6**	
1.6	SQL; SELECT correct fields(1) with userfriendly names FROM two table(1) link table with WHERE Condition(1) '(Age > 30 AND Age < 45) (1) AND (PollutionRiskLevel = "MEDIUM" OR (1) PollutionRiskLevel = "LOW")' ;(1)	**6**	
1.7	SQL: 'SELECT fields(1) do calculation (1) create new field name(1) display with currency(1) from correct table(1) :	**5**	
1.8	SQL: 'SELECT Count(*) (1) AS [Total number of Patients with heart conditions]' (1)'FROM correct table (1) 'WHERE CdtnType Like "Heart%" '; (1)	**4**	
1.9	SQL: SELECT correct fields (1) FROM two tables(1) Join table with WHERE (1) Conditions:Town NOT Like "Johannesburg") (1) AND (PollutionRiskLevel = "HIGH" (1) OR PollutionRiskLevel = "SEVERE")' ; (1)	**6**	
	TOTAL	**40**	

DELPHI SOLUTION QUESTION 1

```
var
  frmDiseases: TfrmDiseases;

implementation

{$R *.dfm}

procedure TfrmDiseases.btnselectAllClick(Sender: TObject);                    //1.1
begin
  qryDiseases.Active := False;            ✓      ✓
  qryDiseases.SQL.Text :=  'SELECT * FROM ConditionsTb ORDER BY CdtnName' ; ✓
  qryDiseases.Active := true;
end;                                                                          (3)
//-------------------------------------------------------------------------------------------------------------------------
procedure TfrmDiseases.btnSelectClick(Sender: TObject);                       //1.2
begin
  qryDiseases.Active := False;          ✓                               ✓
  qryDiseases.SQL.Text :=  'SELECT   PatientID,CdtnName,CdtnType, Age
               FROM ConditionsTb  WHERE Age > 50 ✓ AND CdtnType LIKE "Lung%" ' ;   ✓
  Diseases.Active := true;
end;                                                                          (4)
//-------------------------------------------------------------------------------------------------------------------------
procedure TfrmDiseases.btnUpdateClick(Sender: TObject);                       //1.3
begin
  qryDiseases.Active := False;          ✓                               ✓
  qryDiseases.SQL.Text := 'UPDATE WorkplacesTb SET PollutionRiskLevel = "SEVERE"
                                   WHERE PollutionRiskLevel = "HIGH"'; ✓
  qryDiseases.ExecSQL;
  qryDiseases.SQL.Text := 'SELECT * FROM WorkplacesTb';
  qryDiseases.Active := true;
end;                                                                          (3)
//-------------------------------------------------------------------------------------------------------------------------

procedure TfrmDiseases.btnInsertClick(Sender: TObject);                       //1.4
begin
  qryDiseases.Active := False;   ✓                 ✓
  qryDiseases.SQL.Text := 'INSERT INTO WorkplacesTb ‘ +             ✓
                           ‘ VALUES ("Fac0128", "Factory","Sasolburg", "High")';
  qryDiseases.ExecSQL;
  qryDiseases.SQL.Text := 'SELECT * FROM WorkplacesTb';
  qryDiseases.Active := True;
end;                                                                          (3)
//-------------------------------------------------------------------------------------------------------------------------
procedure TfrmDiseases.btnConditionClick(Sender: TObject);                    //1.5
var
  monthNum    :string;
  con         :string;
begin
  monthNum := InputBox('Enter the number of the month admitted ', '', '');  ✓ // input
  con := InputBox('Enter the condition (such as Lung) ', '', '');

  qryDiseases.Active := False;
                                          ✓
  qryDiseases.SQL.Text := 'SELECT  PatientID, CdtnName, CdtnType, DateAdmitted ' +
                          'FROM ConditionsTb ' +✓      ✓             ✓
```

```
                         'WHERE CdtnType =   " ' + con + ' " AND MONTH(DateAdmitted) =
                         " ' +monthNum +'"  '; ✓
  qryDiseases.ExecSQL;
  qryDiseases.Active := true;
end;                                                                         (6)
//----------------------------------------------------------------------------------
procedure TfrmDiseases.btnMiddleAgeClick(Sender: TObject);                  //1.6
begin
  qryDiseases.Active := False;          ✓                          ✓
  qryDiseases.SQL.Text := 'SELECT PatientID, Age, WorkType AS [Type of workplace],
                                              PollutionRiskLevel ' +
                 'FROM ConditionsTb, WorkPlacesTb '+  ✓
                 'WHERE ConditionsTb.WorkPlaceID = WorkplacesTb.WorkPlaceID  ✓ AND '+
                 '(Age > 30 AND Age < 45)  ✓ AND (PollutionRiskLevel = "MEDIUM" OR
                 PollutionRiskLevel = "LOW")' ; ✓
  qryDiseases.Active := true;
end;                                                                         (6)
//----------------------------------------------------------------------------------
procedure TfrmDiseases.btnSubsidyClick(Sender: TObject);                    //1.7
begin
   qryDiseases.Active := False;           ✓
  qryDiseases.SQL.Text := 'SELECT  PatientID, Age, HoursPerDay,
                                          ✓                 ✓             ✓
                          Format(100 * Age * HoursPerDay, "Currency") AS [Subsidy]' +
                           'FROM ConditionsTb ' ; ✓
   qryDiseases.Active := true;
end;                                                                         (5)
//----------------------------------------------------------------------------------
procedure TfrmDiseases.btnCountHeartClick(Sender: TObject);                 //1.8
begin
  qryDiseases.Active := False;
                                          ✓
  qryDiseases.SQL.Text := 'SELECT Count(*) AS
                             [Total number of Patients with heart conditions]'+ ✓
                           'FROM ConditionsTb '+✓
                            'WHERE CdtnType Like  "Heart%" '; ✓

  qryDiseases.Active := true;
end;                                                                         (4)
//----------------------------------------------------------------------------------
procedure TfrmDiseases.btnNotJhbClick(Sender: TObject);                     //1.9
begin
  qryDiseases.Active := False;                                     ✓
  qryDiseases.SQL.Text := 'SELECT Town, CdtnType AS [Type of Condition], WorkType AS
                                              [Type of work],PollutionRiskLevel '+

            'FROM ConditionsTb, WorkPlacesTb '+✓
            'WHERE ConditionsTB.WorkPlaceID = WorkPlacesTB.WorkPlaceID  ✓ AND '  +
             '(Town NOT Like "Johannesburg")  ✓ AND (PollutionRiskLevel = "HIGH" ✓ OR
            PollutionRiskLevel = "SEVERE")' ; ✓
  qryDiseases.Active := true;
end;                                                                         (6)
//----------------------------------------------------------------------------------


end.
```

QUESTION TWO: DELPHI OOP PROGRAMMING

Mark Allocation

Question Two - Marking Grid			
Question	**Aspect**	**Max Marks**	**Learner's Marks**
2.1	**Object class**		
2.1.1	Declare attributes and methods (subtract marks for errors, max 4) **(4/2=2)**	**2**	
2.1.2	**Constructor:** Parameters: correct order(1) correct data type(1) initialise attributes(2) **(4/2=2)**	**2**	
2.1.3	**toString** method:Definition returns type string(1), in code return string(1), Name with calculated spaces (any acceptable way)- column(2), all the other values as strings(1), labels(1) **(6/2=3)**	**3**	
2.1.4	**Pollutionfacto**r method: returns a value in definition(1) returns value(1), formula correct(2) **(4/2=2)**	**2**	
2.1.5	**Highest Pollutant** method: Definition correct(1), Initialise level(1), if test Co2(1), assign new level(1) and highest pollutant(1) inside if(1), if test lead in the same way(2), if test Mercury in the same way(1), return pollutant(1) **(10/2=5)**	**5**	
2.1.6	**setInfo method**: receive three values(3) Assign the three values to the instance fields of the object(3) **(6/2=3)**	**3**	
2.1.7	**getName** method: Definition correct(1), return name(1) **(2/2=1)**	**1**	
	Subtotal:	**[18]**	
	Main class		
2.2.1	Read from file: Declare array of objects(1), AssignFile(1), Reset file(1), initialise count(1), while not eof (1), inc count(1), read line (1), get position of #(1), copy Name(1), delete(1) Repeat for the three values(3) Instantiate object with parameters(1) Assign to array(1), Close File(1) **(16/2=8)**	**8**	
2.2.2	Menu options: **Display info**: heading(1),for loop(1), call toString method(1) for object(1) **(4/2=2)**	**2**	
	Display Pollution Info: Heading(1), subheadings(1), initialise total(1)for loop(1), work with object(1) call and display getName(1), PollutionFactor(1), add to total(1) calculate average(1), display average outside loop(1) **(10/2=5)**	**5**	
	New Info:Ask name of comp(1), Initalise counter(1) and boolean(1), while loop(2), begin(1), get name from object(1), compare names(1), begin (1), if found, change boolean to true(1), ask other values (3), call set method(1) of object(1), inc counter(1)outside loop, if not found(1) message(1) else(1) update message(1) **(20/2=10)**	**10**	
	Subtotal:	**[25]**	
	TOTAL	**[43]**	

<u>DELPHI SOLUTION QUESTION 2</u>

Object Class

```
unit Company;

interface
uses sysUtils;
  type
  TCompany = class
  private
    CompName          :String;
        Co2           :integer;
        Pb            :integer;   ✓✓
        Hg            :integer;
    public
    constructor Create;  overload;
    constructor Create(nName:String; carbonD, lead, mercury:integer); overload;
    function getPollutionFactor:integer;
    function getHighestPollutant:String;          ✓✓
    function getName:string;
    function toString:string;                                        (4/2=2)
    procedure setName(name:String);
  end;

var
   comp  :TCompany; ✓
   factor:integer;

implementation

  constructor TCompany.Create;
  begin
     CompName := '';
     Co2 := 0;
     Pb  := 0;
     Hg  := 0;
  end;

constructor TCompany.Create(nName:String; carbonD, lead, mercury:integer); ✓
  begin
     CompName := nName;
     Co2 := carbonD;
     Pb  := lead;        ✓✓
     Hg  := mercury;
   end;                                                              (4/2=2)
//---------------------------------------------------------------------------
   function TCompany.getPollutionFactor:integer; ✓
   begin
     factor := Co2 + (2*Pb) + (3*Hg); ✓✓
     getPollutionFactor := factor; ✓
   end;                                                              (4/2=2)
//---------------------------------------------------------------------------
 function TCompany.getHighestPollutant:String; ✓
   var
       level       :integer;
       pollutant  :String;
   begin

      level := 0; ✓
    if Co2 > level then✓
        begin
                level := Co2; ✓
                pollutant := 'Carbon Dioxide'; ✓
         end;
    if pb > level then✓
        begin
                level := pb;            ┐
                pollutant := 'Lead';    ┘            ✓
        end;
    if hg > level then✓
        begin
                level := hg;            ┐
                pollutant := Mercury';  ┘            ✓
        end;
     result := pollutant; ✓
   end;                                                                    (10/2=5)
//-------------------------------------------------------------------------------------
   function TCompany.toString:String ; ✓
   begin
     ✓ toString := CompName + #9 ✓+ 'Carbon Dioxide: ' + IntToStr(Co2) ✓+ #9 + 'Lead  : ' +
                                         IntToStr(Pb) ✓+ #9 + 'Mercury : ' + IntToStr(Hg); ✓
   end;                                                                    (6/2=3)
//-------------------------------------------------------------------------------------
   function TCompany.getName:string; ✓
     begin
       getName := CompName; ✓
     end;                                                                  (2/2=1)
//-------------------------------------------------------------------------------------
procedure TCompany.setInfo(newCo2, newLead, newMerc:integer); ✓✓✓
     begin
        Co2 := newCo2; ✓
        Pb := newLead; ✓
        Hg := newMerc; ✓
     end;
end.                                                                       (6/2=3)
                                                                                  [18]
//==============================================================================
```

DELPHI: Test Class

```
unit Company_U;

interface

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

type
  TfrmPollution = class(TForm)
    redOutput: TRichEdit;
    MainMenu1: TMainMenu;
    Companies1: TMenuItem;
    Listcompaniesdetails1: TMenuItem;
    PollutantsDetails1: TMenuItem;
    Quit1: TMenuItem;
    NewInfo1: TMenuItem;
    procedure Quit1Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure Listcompaniesdetails1Click(Sender: TObject);
    procedure PollutantsDetails1Click(Sender: TObject);
    procedure NewInfo1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmPollution: TfrmPollution;

implementation

uses Company;
var
   arrComp :array[1..20] of TCompany; ✓
   iCount :integer;

{$R *.dfm}

procedure TfrmPollution.Quit1Click(Sender: TObject);
begin
  Application.Terminate;
end;

procedure TfrmPollution.FormActivate(Sender: TObject);
var
  TextF                       :TextFile;
  oneLine, cName, worstName :String;
  iHash, k, factor, Highest    :integer;
  arr                          :array[1..4] of integer;

begin
  redOutput.Paragraph.TabCount := 5;
  redOutput.Paragraph.Tab[1] := 100;
  redOutput.Paragraph.Tab[2] := 150;
  redOutput.Paragraph.Tab[3] := 200;
  redOutput.Paragraph.Tab[4] := 250;
  redOutput.Paragraph.Tab[5] := 300;

  AssignFile(TextF, 'Pollution.txt'); ✓
  if fileExists('Pollution.txt') <> true then
   begin
         ShowMessage('File does not xist');
         Exit;
   end;
   iCount := 0; ✓
   Reset(TextF); ✓
   While not eof(TextF) do✓
     begin
         inc(iCount); ✓
         readln(TextF, oneLine); ✓
         iHash := pos('#',oneLine); ✓
         cName := copy(oneline,1,iHash-1); ✓
```

```
        delete(oneline, 1, iHash); ✓
        for k := 1 to 2 do
         begin
               iHash := pos('#',oneLine);
               arr[k] := StrToInt(copy(oneline,1,iHash-1));      ✓✓✓
               delete(oneline, 1, iHash);
        end;
        arr[3]:= StrToInt(oneLine);

        arrComp[iCount] ✓ := TCompany.Create(cName,arr[1], arr[2], arr[3]); ✓
     end;
  CloseFile(TextF); ✓

end;                                                                    (16/2=8)
//-----------------------------------------------------------------------------------------------------------------------
procedure TfrmPollution.ListCompaniesdetails1Click(Sender: TObject);

var
  K : integer;
  rTotal :real;
begin

  redOutput.Clear;
  redOutput.Lines.Add('List of Companies'); ✓
  redOutput.Lines.Add('==============');
  For K := 1 to iCount do✓
    begin
        redOutput.Lines.Add(arrComp[K].toString); ✓✓
    end;
    redOutput.Lines.Add(' ');
end;                                                                         (4/2=2)
//-----------------------------------------------------------------------------------------------------------------------
procedure TfrmPollution.PollutantsDetails1Click(Sender: TObject);
var
  K, rTotal, pFactor :integer;
begin
  redOutput.Clear;
  redOutput.Lines.Add('Pollutant details of Companies'); ✓
  redOutput.Lines.Add(' ');
  rTotal := 0; ✓
  redOutput.Lines.Add('Company' + #9 + 'Pollutionfactor' + #9 + 'HighestPollutant'); ✓
  redOutput.Lines.Add('=========================================================');
  For K := 1 to iCount do✓
  begin
    with arrComp[K] do✓
     begin
        pFactor :=  getPollutionFactor ;                       ✓            ✓
        redOutput.Lines.Add(getName + #9 + intToStr(pFactor) + #9 +#9 +getHighestPollutant);
        rTotal := rTotal + pFactor; ✓
      end;
   end;
   redOutput.Lines.Add(' ');
   redOutput.Lines.Add('Average pollutionfactor is '+ FloatToStrF(rTotal/iCount✓, ffFixed,6,2)); ✓

end;                                                                        (10/2=5)
//-----------------------------------------------------------------------------------------------------------------------
procedure TfrmPollution.NewInfo1Click(Sender: TObject);
```

```
var
  sName                        :string;
  k ,newCo2, newLead, newMerc  :integer;
  found                        :boolean;
begin
  sName := InputBox('Name of Company? ', '', ''); ✓
  k := 0; ✓
  while (k < iCount) ✓and (found = false) ✓do
    begin✓
         inc(k); ✓
          if arrComp[k].getName ✓= sName then✓
           begin✓
                 found := true; ✓
                 newCo2 := StrToInt✓ (InputBox('Level of Co2 ?', '', '')); ✓
                 newLead := StrToInt(InputBox('Level of Lead ?', '', '')); ✓
                 newMerc := StrToInt(InputBox('Level of Mercury ?', '', '')); ✓
                 arrComp[k].setInfo✓ (newCo2, newLead, newMerc); ✓
           end;
     end;
    if not(found) then✓
         showMessage(sName + ' not found') ✓
    else✓
         showMessage(sName + ' Info updated'); ✓
end;                                                              (20/2=10)

end.                                                                  [25]
//=====================================================================
```

Subtotal Question2: [43]

QUESTION THREE: DELPHI

Mark Allocation

Question Three - Marking Grid			
Question	**Aspect**	**Max Marks**	**Learner's Marks**
3.1	Declare array(1) **ClearSea button**: Two loops(2), assign character (1) Clear stgGrid: 2 loops(2), assign space(1), clear RichEdit(1) **(8/2 = 4)**	**4**	
3.2	**DisplayGrid method:** Heading (1) Column headings(1) Row labels(1) two for loops(1) using size of array(1), display content (1) Each row on a new line (1) Grid correct size(1) **(8/2 = 4)**	**4**	
3.3	**Definition of function**: three variables(1) correct order(1), integer(1), return value Boolean(1), Boolean true outside if (1) If to test boundaries(3) value false (1) return value(1) **(10/2 = 5)**	**5**	
3.4 **3.4.1**	Randomise(1) Request user friendly(1) the size of grid(1) input size(1) Convert to int(1)call validate function(1) in loop(1) Call clearSea procedure(1) **(8/2=4)**	**4**	
3.4.2	Request user friendly the seriousness of spill(1) Call validate function(1) in loop to test spill(1) input value(1) Initialise count (1), Calculate number spots to pollute (1) while loop(1) random x and y generated(2) If not "+" character(1), allocate + character (1) increment count(1) inside if(1) Call display procedure(1) **(14/2 = 7)**	**7**	
3.5	Call procedures clearSea and DisplayGrid in Clear the Sea button **(2/2 = 1)**	**1**	

3.6
3.6.1
	repeat (1) request user friendly (1)
	Loop(1) input x, (1) convert to int (1) Loop(1) input y (1) convert to int(1) **(8/2=4)**
		4	

3.6.2	Identify high risk(2) else(1) Begin(1)Initialise count(1) Two for loops covering the correct rows and columns(2) If to exclude the spots outside the grid(1) If to count the + characters(1) begin (1) increment count(1) If to identify Risky area (1) message(1) identify Low risk(1) message(1) Display message(1) **(16/2 = 8)**	**8**	
	TOTAL	**[37]**	

DELPHI SOLUTION QUESTION 3

```
unit OilSpill_U;

interface

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

type
  TfrmOilSpill = class(TForm)
    stgOilSpill: TStringGrid;
    redOutput: TRichEdit;
    Panel1: TPanel;
    btnOilSpill: TButton;
    btnEvaluate: TButton;
    BitBtn2: TBitBtn;
    lblHeading: TLabel;
    lblStgHeading: TLabel;
    btnClear: TButton;
    Procedure ClearSea;
    Procedure DisplayGrid;
    procedure btnOilSpillClick(Sender: TObject);
    procedure btnEvaluateClick(Sender: TObject);
    procedure btnClearSeaClick(Sender: TObject);
    private     { Private declarations }


  public
    { Public declarations }

  end;
var
  frmOilSpill: TfrmOilSpill;


implementation
var
  TwoD           :array[1..20,1..20] of char; ✓
  Size           :integer;

{$R *.dfm}
//-----------------------------------------------------------------------------------------------------------------------
Procedure TfrmOilSpill.ClearSea;
var
  R, C  :integer;
begin
  stgOilSpill.Visible := true;
   for R := 1 to 20 do✓
      for C := 1 to 20 do✓
         TwoD[R,C] := '-'; ✓

   for R := 0 to 20 do✓
      for C := 0 to 20 do✓
         stgOilSpill.Cells[C, R] ✓ := ' '; ✓
   redOutput.Clear; ✓
```

```
end;                                                                     (8 / 2) = 4

//-------------------------------------------------------------------------------------------------------------------
procedure TfrmOilSpill.DisplayGrid;
var
  R, C  :integer;
begin
   lblStgHeading.Caption := 'Oil spill on the open sea'; ✓
   for C := 1 to size do✓
      stgOilSpill.Cells[C, 0] := IntToStr(C); ✓
   for R := 1 to size do✓
       stgOilSpill.Cells[0, R] := IntToStr(R); ✓

   for R := 1 to size do✓
      for C := 1 to size do✓
         stgOilSpill.Cells[C, R] := TwoD[R, C]; ✓
end;                                                                     (8 / 2) = 4
//-------------------------------------------------------------------------------------------------------------
function Validate(LowB, HighB, value:integer):boolean; ✓ 3 parameters✓ correct order✓integer✓

begin
  Validate✓ := true; ✓
    if (value < LowB) ✓ or✓ (value > HighB) ✓ then
           Validate := false; ✓
end;                                                                     (10 / 2 = 5)
//-------------------------------------------------------------------------------------------------------------------
procedure TfrmOilSpill.btnOilSpillClick(Sender: TObject);
var
  iCount, iNumber, iRandomX, iRandomY, R, C, iSpillLevel :integer;
begin
  Randomize; ✓
 repeat✓
   Size := StrToInt✓ (InputBox✓ ('Size of the grid', 'How big is grid for the simulation? (10 - 20)
✓',''));
 until ✓Validate(10,20,Size); ✓

ClearSea; ✓                                                              (8/2 = 4)

 repeat✓
     iSpillLevel := StrToInt(InputBox('How serious is the spill?', 'Type in a number in the range (1-
10)','')); ✓
   until Validate(1, 10, iSpillLevel); ✓

  iNumber := 10 * iSpillLevel; ✓
  iCount := 0; ✓
  while iCount < iNumber do✓
    begin
      iRandomX := Random(Size) ✓+1; ✓
      iRandomY := Random(Size)+1; ✓
      if TwoD[iRandomX, iRandomY] <> '+' then✓
        begin✓
          inc(iCount); ✓
          TwoD[iRandomX, iRandomY] := '+'; ✓
        end;
    end;
    DisplayGrid; ✓
    btnEvaluate.Enabled :=true; ✓
```

```
end;                                                                    (14/2 = 7)
//-----------------------------------------------------------------------------------------------------------
procedure TfrmOilSpill.btnClearSeaClick(Sender: TObject);
begin
  ClearSea; ✓
  DisplayGrid; ✓
end;                                                                    (2/2 = 1)
//-----------------------------------------------------------------------------------------------------------
procedure TfrmOilSpill.btnEvaluateClick(Sender: TObject);
var
 iRowTotal,iRow, iCol, R, C, iCount, xPos, yPos:integer;
 sMessage:string;
begin

 repeat✓
   xPos := StrToInt✓ (InputBox✓ ('Rows', 'Type in a X value for the position, 1 to ' + IntToStr(size)
✓,''));
 until Validate(0, size, xPos); ✓

 repeat✓
   yPos := StrToInt(InputBox('Rows', 'Type in a Y value for the position, 1 to ' + IntToStr(size),'')); ✓
 until Validate(0, size, yPos); ✓


 if TwoD[xPos, yPos] = '+' then✓
   sMessage := 'High risk in position ' + IntToStr(xPos) + ',' + IntToStr(yPos) ✓
 else✓
  begin✓
   iCount := 0; ✓

    for R := xPos-1 to xPos + 1 do✓
      for C := yPos - 1 to yPos + 1 do✓
        if ( xPos >=1 )and (yPos >= 1) then✓
          if(xPos <= size) and (yPos <= size) then✓
            if TwoD[R,C] = '+' then✓
              inc(iCount); ✓

     if iCount >= 4 then✓
       sMessage := 'Risky area in position ' + IntToStr(xPos) + ',' + IntToStr(yPos) ✓
     else✓
       sMessage := 'Low risk area in position ' + IntToStr(xPos) + ',' + IntToStr(yPos); ✓
   end;
 redOutput.Lines.Add(sMessage); ✓
end;                                                                    (24/2 = 12)
//-----------------------------------------------------------------------------------------------------------
end.
```

Total Question 3 : [37]

SECTION B: JAVA PROGRAMMING

QUESTION ONE: JAVA DATABASE CONNECTIVITY

Mark Allocation

Question One - Marking Grid			
Question	**Aspect**	**Max Marks**	**Learner's Marks**
1.1	SQL: Select all fields(1) and display all the fields (1) sorted according to name of condition(1)	**3**	
1.2	SQL: Selected fields (1) from correct table(1) Age > 50 (1), "Lung%" (1)	**4**	
1.3	SQL: Update (1), SET field (1), condition (1)	**3**	
1.4	SQL: Insert (1), table VALUES (1), correct fields(1)	**3**	
1.5	Input(1) SQL: SELECT fields(1) from correct table(1) WHERE CdtnType = " ' + con + ' " (1) AND MONTH(DateAdmitted)(1) = " ' +monthNum +'" (1)	**6**	
1.6	SQL; SELECT correct fields(1) with userfriendly names FROM two table(1) link table with WHERE Condition(1) '(Age > 30 AND Age < 45) (1) AND (PollutionRiskLevel = "MEDIUM" OR (1) PollutionRiskLevel = "LOW")' ;(1)	**6**	
1.7	SQL: 'SELECT fields(1) do calculation (1) create new field name(1) display with currency(1) from correct table(1) :	**5**	
1.8	SQL: 'SELECT Count(*) (1) AS [Total number of Patients with heart conditions]' (1)'FROM correct table (1) 'WHERE CdtnType Like "Heart%" '; (1)	**4**	
1.9	SQL: SELECT correct fields (1) FROM two tables(1) Join table with WHERE (1) Conditions:Town NOT Like "Johannesburg") (1) AND (PollutionRiskLevel = "HIGH" (1) OR PollutionRiskLevel = "SEVERE")' ; (1)	**6**	
	TOTAL	**40**	

JAVA SOLUTION   QUESTION 1

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

public class DiseasesDataBase
  {
    Connection conn;


    public DiseasesDataBase ()
    {
    //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:diseases.mdb");

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

        //String filename = inKb.readLine();    // For entering the location of the database

       String filename = "E:/DiseasesDB.mdb"; // this statement has been used to avoid having to
                                             //enter the location of the database every time
                                                     //you run the program.

        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 Diseases database successfully established");

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


  //-----------------------------------------------------------------------------------------------------------------
    public void selectAllQuery ()throws SQLException                                    //1.1
    {
      Statement stmt = conn.createStatement ();
                                       ✓                ✓                 ✓
      String sql = "SELECT * FROM ConditionsTb ORDER BY CdtnName";
      ResultSet rs = stmt.executeQuery (sql);
      System.out.printf("%-12s%-27s%-25s%-15s%-12s%-16s%-20s","PatientID","Condition
        Name","Type of condition","Age","WorkPlaceID"," DateAdmitted","Hours per Days");
     System.out.println();
      System.out.println("==========================================================");
     while (rs.next ())
      {
        String id = rs.getString ("PatientID");
        String cName = rs.getString ("CdtnName");
        String cType = rs.getString ("CdtnType");
        String age = rs.getString ("Age");
        String workID = rs.getString ("WorkPlaceID");
        String date = rs.getString("DateAdmitted");
        date = date.substring(0,10);
        String hours = rs.getString("HoursPerDay");
        System.out.printf("%-10s%-27s%-20s%-8s%-12s%-16s%-20s",id,cName,cType,age,
                                                            workID,date,hours);
        System.out.println();
      }
      System.out.println(" ");
      stmt.close ();
    } //select All                                                                     (3)
  //----------------------------------------------------------------------------------------
     public void selectAgeQuery ()throws SQLException                                //1.2
    {
      Statement stmt = conn.createStatement ();
                                                ✓                           ✓
      String sql = "SELECT PatientID, CdtnName, CdtnType, Age FROM ConditionsTb
                        WHERE age > 50  ✓  AND CdtnType LIKE 'Lung%'";✓
        ResultSet rs = stmt.executeQuery (sql);
      System.out.printf("%-10s%-27s%-20s %-12s","PatientID","Condition Name","Condition
                                                                          Type", "Age");
     System.out.println();
      System.out.println("================================================");
      while (rs.next ())
      {
        String id = rs.getString ("PatientID");
        String cn = rs.getString ("CdtnName");
        String ct = rs.getString ("CdtnType");
        String age = rs.getString ("Age");
        System.out.printf("%-10s%-27s%-20s%-12s",id,cn,ct,age);
        System.out.println();

      }
      System.out.println(" ");
      stmt.close ();
    } //select    Age                                                                  (4)

    //--------------------------------------------------------------------------------------
    public void updateQuery() throws SQLException                                    //1.3
    {
     Statement stmt = conn.createStatement ();
                           ✓
      String sql = "UPDATE WorkPlacesTb                                                    ✓
                SET PollutionRiskLevel = 'SEVERE' ✓ WHERE PollutionRiskLevel = 'HIGH'";

      System.out.println (" Updated ");

      stmt.close();
    }                                                                               (3)
//-------------------------------------------------------------------------------------------------
 public void insertQuery () throws SQLException                                     //1.4
   {

      Statement stmt = conn.createStatement ();
                          ✓            ✓
      String sql = "INSERT INTO WorkPlacesTb              ✓
                          VALUES ( 'Fac012','Factory','Sasolburg','High')";

      int numRows = stmt.executeUpdate (sql);

      System.out.println (" Record inserted");

      stmt.close ();

   }                                                                                (3)
 //-------------------------------------------------------------------------------------------------
   public void getConditionQuery ()throws SQLException                              //1.5
     {
       System.out.println("\f");
       System.out.println();

       Statement stmt = conn.createStatement ();

       String numMonth = JOptionPane.showInputDialog("Type in the number of the month ");

       String con= JOptionPane.showInputDialog("Type in the condition");✓  // input
                                                                        ✓
       String sql = "SELECT  PatientID, CdtnName, CdtnType, DateAdmitted
                          FROM ConditionsTb ✓
                          WHERE CdtnType = '" + con + "'✓ AND
                                 MONTH(DateAdmitted) ✓ = '" + numMonth + "'    "; ✓

       ResultSet rs = stmt.executeQuery (sql);

       System.out.printf("%-10s%-27s%-20s%-12s","PatientID","Condition Name","Condition Type",
                                                                    "Date Admitted");
      System.out.println();
      System.out.println("===========================================================");

       while (rs.next ())
       {
         String id = rs.getString ("PatientID");
         String name = rs.getString ("CdtnName");
          String type = rs.getString ("CdtnType");
         String sDate = rs.getString ("DateAdmitted");
         sDate = sDate.substring(0,10);


         System.out.printf("%-10s%-27s%-20s%-12s",id,name, type,sDate);
```

```
        System.out.println();

      }
      System.out.println(" ");
      stmt.close ();
    }                                                                                    (6)
    //-------------------------------------------------------------------------------------------------------------------------
```

```
    public void getMiddleAgeQuery ()throws SQLException                                  //1.6
    {
      System.out.println("\f");
      System.out.println();

      Statement stmt  = conn.createStatement ();
                                                                    ✓
      String sql = "SELECT PatientID, Age, WorkType AS [Type of workplace],
                                                                    PollutionRiskLevel
                    FROM  ConditionsTb, WorkplacesTb "+✓
               "WHERE ConditionsTb.WorkPlaceID = WorkplacesTb.WorkPlaceID ✓AND
                    Age > 30 AND Age < 45 ✓AND (PollutionRiskLevel = 'MEDIUM' ✓OR
                                     PollutionRiskLevel = 'LOW')" ; ✓
      ResultSet rs = stmt.executeQuery (sql);
      System.out.printf("%-10s%-10s%-15s%-15s","PatientID","Age","WorkType","
                                                                    PollutionRiskLevel");
      System.out.println();

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

        String id = rs.getString ("PatientID");
        String age = rs.getString ("Age");
        String wplace = rs.getString ("Type of workplace");
        String pollution = rs.getString ("PollutionRiskLevel");
        System.out.printf("%-10s%-10s%-15s%-15s",id,age,wplace,pollution);
        System.out.println();
      }

     System.out.println(" ");
      stmt.close ();
    }                                                                                    (6)
//-------------------------------------------------------------------------------------------------------------------------
```

```
public void getSubsidyQuery()throws SQLException                                         //1.7
    {
      System.out.println("\f");
      System.out.println();

      Statement stmt = conn.createStatement ();
                                                       ✓
      String sql = "SELECT  PatientID, Age, HoursPerDay,
                    Format(100 * Age * HoursPerDay, ✓ 'Currency') ✓ AS [Subsidy] ✓
                     FROM ConditionsTb " ; ✓

      ResultSet rs = stmt.executeQuery (sql);
      System.out.printf("%10s%10s%15s%15s","PatientID","Age", "HoursPerDay","Subsidy");
      System.out.println();
      System.out.println("=========================================================");
```

```
     while (rs.next ())
     {
         String id = rs.getString ("PatientID");
        String age = rs.getString ("Age");
        String hours = rs.getString ("HoursPerDay");
        String subsidy = rs.getString("Subsidy");
        System.out.printf("%10s%10s%10s%22s",id,age,hours,subsidy);
        System.out.println();
     }
     System.out.println(" ");
     stmt.close ();
  }                                                                                    (5)
 //-------------------------------------------------------------------------------------------
  public void countHeartQuery()throws SQLException                                     //1.8
  {
      System.out.println("\f");
      System.out.println();

      Statement stmt = conn.createStatement ();
                                         ✓       ✓             ✓
      String sql = "SELECT Count(*) AS [Total] FROM ConditionsTb "+
                 "WHERE CdtnType Like  'Heart%' " ; ✓

      ResultSet rs = stmt.executeQuery(sql);
      System.out.println();
      System.out.println();
      while (rs.next ())
     {
      String num = rs.getString ("Total");

      System.out.println("Number of patients with heart conditions: " + num);
    }
     System.out.println();
     System.out.println();
     stmt.close ();
  }                                                                                    (4)
//-------------------------------------------------------------------------------------------
public void getNOTJhbQuery()throws SQLException                                        //1.9
  {
     System.out.println("\f");
     System.out.println();

     Statement stmt = conn.createStatement ();
                                         ✓
     String sql = "SELECT Town, CdtnType AS [Type of Condition],
                WorkType AS [Type of Work] ✓
                FROM ConditionsTb, WorkplacesTb ✓
                WHERE ConditionsTb.WorkPlaceID = WorkplacesTb.WorkPlaceID " +
                " AND (Town NOT Like 'Johannesburg')    AND (PollutionRiskLevel = 'HIGH'
                                              OR  PollutionRiskLevel = 'SEVERE')" ;
     ResultSet rs = stmt.executeQuery (sql);
     System.out.printf("%-28s%-20s%-15s","Town","CdtnType", "WorkType");
     System.out.println();
     System.out.println("=======================================================");
     while (rs.next ())
     {
        String town = rs.getString ("Town");
        String con = rs.getString("Type of Condition");
        String work = rs.getString("Type of Work");
        System.out.printf("%-28s%-20s%-15s",town,con,work);
        System.out.println();
      }
      System.out.println(" ");
      stmt.close ();
    } // Not JHB                                                                    (6)
//------------------------------------------------------------------------------------------
public void disconnect () throws SQLException
    {

      conn.close ();
    }

}
```

Subtotal Question 1 : [40]

QUESTION 2 JAVA

Mark Allocation

Question Two - Marking Grid			
Question	**Aspect**	**Max Marks**	**Learner's Marks**
2.1	**Object class**		
2.1.1	Declare attributes and methods (subtract marks for errors, max 4) **(4/2=2)**	**2**	
2.1.2	**Constructor:** Parameters: correct order(1) correct data type(1) initialise attributes(2) **(4/2=2)**	**2**	
2.1.3	**toString** method:Definition/heading returns type string(1), in code return string(1), Name with calculated spaces(any acceptable way) - column(2), all the other values as strings(1), labels(1) **(6/2=3)**	**3**	
2.1.4	**Pollutionfacto**r method: returns a value in definition(1) returns value(1), formula correct(2) **(4/2=2)**	**2**	
2.1.5	**Highest Pollutant** method: Definition correct(1), Initialise level(1), if test Co2(1), assign new level(1) and highest pollutant(1) inside if(1), if test lead in the same way(2), if test Mercury in the same way(1), return pollutant(1) **(10/2=5)**	**5**	
2.1.6	**setInfo method**: receive three values(3) Assign the three values to the instance fields of the object(3) **(6/2=3)**	**3**	
2.1.7	**getName** method: Definition correct(1), return name(1) **(2/2=1)**	**1**	
	Subtotal:	**[18]**	
2.2.1	**testCompany class** Read from file: Declare array of objects(1), Init bufferedReader(1), initialise count(1), readline(1)while not null (1), use position of #(1), copy Name(1), and the three values(3) Instantiate object with parameters(2) Assign to array(1), inc count(1) read line (1), Close File(1) **(16/2=8)**	**8**	
2.2.2	**Menu options**:		
	Display info: heading(1),for loop(1), call toString method(1) for object(1) **(4/2=2)**	**2**	
	Display Pollution Info: Heading(1), subheadings(1), initialise total(1)for loop(1), work with object(1) call and display getName(1), PollutionFactor(1), add to total(1) calculate average(1), display average outside loop(1) **(10/2=5)**	**5**	
	New Info:Ask name of comp(1), Initalise counter(1) and boolean(1), while loop(2), begin(1), get name from object(1), compare names(1), begin (1), if found, change boolean to true(1), ask other values (3), call set method(1) of object(1), inc counter(1)outside loop, if not found(1) message(1) else(1) update message(1) **(20/2=10)**	**10**	
	Subtotal:	**[25]**	
	TOTAL	**[43]**	

JAVA SOLUTION   QUESTION 2

```
 public class Company
 {
    private String name = "";✓
    private int co2 = 0; ✓
    private int pb = 0;  ✓
    private int hg = 0;✓                                                          (4/2=2)

   public Company()
    {
    }

   public Company (String name, int co2, int pb, int hg) ✓✓
    {
      this.name = name;
      this.co2 = co2;           } ✓✓
      this.pb = pb;
      this.hg = hg;
    }                                                                             (4/2=2)
//--------------------------------------------------------------------------------------------
     public int ✓getPollutionFactor()
    {
      int factor = co2 + 2 * pb + 3 * hg; ✓✓
      return factor; ✓
    }                                                                             (4/2=2)
//--------------------------------------------------------------------------------------------
     public String ✓getHighestPollutant()
    {
      String pollutant = "";
      int level = 0; ✓
      if (co2 > level) ✓
      {
        level = co2; ✓
        pollutant = "Carbon Dioxide";  ✓
      }
      if (pb > level) ✓
      {
        level = pb;             } ✓
        pollutant = "Lead";
      }
      if (hg > level) ✓
      {
        level = hg;             } ✓
        pollutant = "Mercury";
      }
      return pollutant; ✓
    }                                                                             (10/2=5)
//--------------------------------------------------------------------------------------------
public String spaces(String s,int n)
    {
      String spc = "";
      for (int i = 0; i < n - s.length(); i++)
      {
        spc +=" ";
      }
      return spc;
    }
//------------------------------------------------------------------------------------------------------------------------
  public String toString()
    {

       String s = name + spaces(name,20) ✓+ "\tCarbon Dioxide: " + co2 +✓ "\tLead: " + pb + ✓
                                                                   ✓"\t\tMercury: " + hg; ✓
      return s; ✓
    }                                                                           (6/2=3)
   //------------------------------------------------------------------------------------------------------------------
    public void ✓setInfo(int newCo2, int newLead, int newMerc) ✓✓
    {
       this.co2= newCo2; ✓
       this.pb = newLead; ✓
       this.hg = newMerc; ✓
    }                                                                           (6/2=3)
//----------------------------------------------------------------------------------------------------------------------
    public String getName()✓
    {
       return name; ✓
    }
}                                                                              (2/2 = 1)
                                                                                    [18]
//---------------------------------------------------------------------------------------------------------------------------
```

// testCompany class

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

   public class testCompany
  {

     public static void main(String args[]) throws Exception✓
    {
       BufferedReader in = new BufferedReader (new InputStreamReader (System.in));
       BufferedReader fr = new BufferedReader (new FileReader ("Pollution.txt"));✓
       int count = 0; ✓
       Company [ ] arrComp = new Company[20]; ✓
       String line = fr.readLine();✓

       while ( line!=null ) ✓
       {

          String [ ] part = line.split("#");✓
          String sName = part[0]; ✓
          int co2 = Integer.parseInt(part[1]); ✓
          int pb = Integer.parseInt(part[2]); ✓
          int hg = Integer.parseInt(part[3]); ✓

          arrComp[count] ✓= new Company(sName,co2,pb,hg); ✓

         line = fr.readLine();✓
         count++;✓
       }
       fr.close();✓                                                              (16/2=8)
       //-----------------------------------------------------------------------------------------------------------
       BufferedReader inKb = new BufferedReader (new InputStreamReader (System.in));
       char ch = ' ';

       while (ch != 'Q')
       {
          System.out.println("\n\n");
          System.out.println("A - List All Companies");
          System.out.println("B - Pollutants Details");
          System.out.println("C - New Info");
          System.out.println("Q - QUIT");

          System.out.print("Your Choice? :");
          ch = inKb.readLine().toUpperCase().charAt(0);
          switch (ch)
          {
            case 'A':
              {
                //System.out.println("\f");
                System.out.println("List of All Companies");✓
                System.out.println("====================");
                for (int k = 0; k < count; k++)✓
                {
                   System.out.println(arrComp[k].toString());✓✓

                }
                break;                                                        (4/2=2)
              } //---------------------------------------------------------------------------------------------------------------
            case 'B':
              {
                double total = 0; ✓
                //System.out.println("\f");
                System.out.println("List of pollutant detail");✓
                System.out.println("");
                System.out.printf("%-20s%-20s%-20s","Company","Pollution Factor","Highest
                                                                              pollutant");✓
                System.out.println("");

                System.out.println("================================================");
                for (int k = 0; k < count; k++)✓
                {
                   int pFactor = arrComp[k].getPollutionFactor();✓
                   System.out.printf("%-20s%-20s%-20s",arrComp[k].getName(),✓
                                                  pFactor,arrComp[k].getHighestPollutant());✓
                   System.out.println(" ");
                   total = total + pFactor; ✓
                }
                System.out.println("");
                double ave = total/count; ✓
                System.out.printf("%-20s%10.2f","The average pollution factor is ",ave); ✓
                System.out.println("");
                break;                                                        (10/2=5)
              } //-----------------------------------------------------------------------------------------------------------------

            case 'C':
              {
                String sCompName = JOptionPane.showInputDialog("Company name?: ");✓
                int k = 0; ✓
                boolean found = false; ✓
```

```
           while (!(found) ✓ && (k < count)) ✓
           {
             String sName = arrComp[k].getName();✓
              if (sName.equalsIgnoreCase(sCompName)) ✓✓
             {
                found = true; ✓
                int newCo2 = Integer.parseInt✓ (JOptionPane.showInputDialog(
                                              "Type in the new carbon dioxide level "));✓
                int newLead = Integer.parseInt(JOptionPane.showInputDialog(
                                              "Type in the new lead level "));✓
                int newMerc = Integer.parseInt(JOptionPane.showInputDialog(
                                              "Type in the new mercury level "));✓
                arrComp[k].setInfo(newCo2, newLead, newMerc); ✓✓
             }
            k++;✓
           }
           if (found) ✓
             System.out.println(sCompName + " updated");✓
           else✓
             System.out.println(sCompName + " not found");✓
           break;                                                          (20/2=10)
          }//-----------------------------------------------------------------------

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

       } //while
     }  //main

   } //class                                                                   [25]
 }//----------------------------------------------------------------------------------
```

Subtotal: Question 2: [43]

QUESTION 3 JAVA PROGRAMMING

Mark Allocation

Question Three - Mark Grid			
Question	**Aspect**	**Max Marks**	**Learner's Marks**
3.1	Declare array(1) **clearSea method**: Two loops(2), assign character (1) **(4/2 = 2)**	**2**	
3.2	**DisplayGrid method:** Heading (1) spacing before column headings(1) Column headings(1) Row labels & spacing(2) two for loops(2) using size of array(1), display content (1) Each row on a new line (1) Spacing between characters(1) Grid correct size(1) **(12/2 = 6)**	**6**	
3.3	**Validate method**: Definition of method: three variables(1) correct order(1), integer(1), return value Boolean(1), Boolean true outside if (1) If to test boundaries(3) value false (1) return value(1) **(10/2 = 5)**	**5**	
3.4 **3.4.1**	throws IOException (1) Initialise bufferedReader(1) Request user friendly(1) the size of grid(1) input size(1) call validate method(1) in loop(2) **(8/2=4)**	**4**	
3.4.2	Call clearSea method(1) Request user friendly the seriousness of spill(1) Call validate method in loop to test spill(1) input value(1) Initialise count (1), Calculate number spots to pollute (1) while loop(1) random x and y generated(2) If not "+" character(1), allocate + character (1) increment count(1) inside if(1) Call display method(1) **(14/2 = 7)**	**7**	
3.5	Call methods clearSea and DisplayGrid in Clear the Sea option in test class **(2/2 = 1)**	**1**	
3.6 **3.6.1**	throws IOException (1) and Initialise BufferedReader(1) Request user friendly Loop(1) input x, (1) decrement x (1) Loop(1) input y (1) decrement y(1) **(8/2 = 4)**	**4**	
3.6.2	Identify high risk(2) else(1) Initialise count(1) Two for loops covering the correct rows and columns(2) If to exclude the spots outside the grid(1) If to count the + characters(1) increment count(1) If to identify Risky area (1) message(1) identify Low risk(1) message(1) Display message(1) Ask next evaluation(1) while(1) **(16/2 = 8)**	**8**	
	TOTAL	**[37]**	

JAVA SOLUTION QUESTION 3

```
import java.io.*;
public class OilSpill
{
  private int size = 20; ✓
  private int spillLevel; ✓
  private char ✓ [ ][ ] twoD;    ✓
                                                          (4/ 2 = 2)

 OilSpill()
  {
  }
  //-----------------------------------------------------------------------------------------------
   public boolean ✓ validate (int value, int lower, int upper) 3 value✓ in correct order✓ integer✓
  {
    boolean valid = true; ✓
    if ((value < lower) ✓||✓ (value > upper)) ✓
    {
      valid = false; ✓
    }
    return valid; ✓
  }                                                       (10/ 2 = 5)
  //-----------------------------------------------------------------------------------------------
   public void displayGrid()
  {
    System.out.println("    Oil Spill on the open sea");✓
    System.out.println(" ");
     System.out.printf("%-3s", " ");  ✓
     for (int c = 0 ; c < size; c++ ) ✓
      {
        System.out.printf ("%-2s ",✓ (c+1)); ✓
      }
      System.out.println();✓

     for (int r = 0; r < size; r++)✓
    {
      System.out.printf("%-3s"✓,(r+1)); ✓
       for (int c = 0 ; c < size; c++ ) ✓
      {
        System.out.printf ("%-3s", twoD[r][c]); ✓
      }
      System.out.println();✓
    }
   System.out.println();
  }                                                       (12/ 2 = 6)
//-------------------------------------------------------------------------------------------------
  public void clearSea()
  {
    this.size = size;

    twoD = new char[size][size]; ✓

    for (int r = 0; r < size; r++)✓
    {
      for (int c = 0 ; c < size; c++ ) ✓
      {
          twoD[r][c] = '-'; ✓
        }
      }

    }                                                                                    (4 / 2 = 2)
  //-------------------------------------------------------------------------------------------------------------------------------
  public void simulateOilSpill() throws Exception ✓
    {
      BufferedReader inKb = new BufferedReader (new InputStreamReader (System.in)); ✓
      do
      {
      System.out.print("How big is grid for the simulation? ✓ (10 - 20)"); ✓
      size =Integer.parseInt✓ (inKb.readLine());✓
      }
      while (! validate(size,10,20) ✓); ✓                                                 (8/2 = 4)

      clearSea();✓
      do
      {
       System.out.print("How serious is the spill  (1-10)?"); ✓
       spillLevel = Integer.parseInt(inKb.readLine());✓
      }
      while (!validate(spillLevel, 1, 10)); ✓

     // place oil spots , no duplicates
     int count = 0; ✓
     int number = 10 * spillLevel; ✓
     while (count < number) ✓
     {

       int xtarget = (int)(Math.random()* size); ✓

       int ytarget = (int)(Math.random()* size); ✓

       if ( twoD[xtarget][ytarget] != '+') ✓
       {✓
         count ++;✓
         twoD[xtarget][ytarget] = '+'; ✓
       }
     }
    displayGrid();✓

  }                                                                                      (14 / 2 = 7)
 //--------------------------------------------------------------------------------------------------------------------------------
  public void evaluatePosition() throws Exception
    {
     String ans;
     displayGrid();
     int xPos = 0;
     int yPos = 0;
     do ✓  // repeat evaluation more than once
     {

      String message = "";
      BufferedReader inKb = new BufferedReader (new InputStreamReader (System.in)); ✓

       do
```

```
        {
          System.out.println("Enter a number for the X position and Y position (1 - " + size + ") ");
          System.out.print("X Pos?");
          xPos = Integer.parseInt(inKb.readLine());✓
          xPos--;✓
        }
       while (!validate(xPos,0,size-1)); ✓

     do
     {
       System.out.print("Y Pos?");
       yPos = Integer.parseInt(inKb.readLine());✓
       yPos--;✓
      }
      while (!validate(yPos, 0, size-1)); ✓


       if (twoD[xPos][yPos] == '+')  ✓
       {
        message = "High risk in position " + (xPos+1) + "," + (yPos+1); ✓
       }
     else✓
    {

     int count = 0; ✓
     for (int r = xPos-1; r <= xPos + 1; r++)✓
      for (int c = yPos – 1; c <= yPos + 1; c++)✓
      {
        if( (r > -1) && (c >-1) && (r < size) && (c < size)) ✓ //provide for x / y on the edge of the grid
        {
          if (twoD[r][c] == '+') ✓
          count++;✓
       }
     }
     if (count > 4) ✓
     {
       message = "Risky area in position " + (xPos+1) + "," + (yPos+1) ; ✓
     }
     else✓
     {
        message = "Low risk area in position " + (xPos+1) + "," + (yPos+1); ✓
     }
    } //else
  System.out.println(message); ✓
  System.out.println();
  System.out.print("Another position (Y/N)");
  ans = inKb.readLine();✓
  }
  while(!ans.equalsIgnoreCase ("N")); ✓
 }
}                                                                              (24 / 2 = 12)
//--------------------------------------------------------------------------------------------------------------------------------
```

Total Question 3: [38]

FINAL TOTAL: 120