Province of the
EASTERN CAPE
EDUCATION

Iphondo leMpuma Kapa: Isebe leMfundo
Provinsie van die Oos Kaap: Department van Onderwys
Porafensie Ya Kapa Botjahabela: Lefapha la Thuto

NATIONAL SENIOR CERTIFICATE

GRADE 12

SEPTEMBER 2025

INFORMATION TECHNOLOGY P1
MARKING GUIDELINE

MARKS: **150**

This marking guideline consists of 19 pages.

NAME OF LEARNER:				
TOTAL QUESTION 1	**TOTAL QUESTION 2**	**TOTAL QUESTION 3**	**TOTAL QUESTION 4**	**TOTAL**
/35	**/40**	**/45**	**/30**	**/150**

	QUESTION 1	MAX MARK	MARKS ACHIEVED
1.1	**BUTTON: [Q1.1 – Sun shape]** Create a TShape object dynamically ✓ Set Parent to pnlOutput ✓ Set Shape to stCircle ✓ Set Width and Height to 50 ✓ Set Brush.Color to clYellow ✓ Set Align to alClient. ✓	**6**	
1.2	**BUTTON: [Q1.2 – Distance conversion]** Read value from edtDistance and convert using StrToFloat ✓ Multiply by correct conversion factor (9.461) ✓ Store result in a suitable real variable ✓ Round to nearest trillion ✓ Output to lblQ1_2.Caption ✓ "[x] ✓ light years is approximately [y] trillion km" ✓ Correct format for both [x] and [y] ✓	**8**	
1.3	**BUTTON: [Q1.3 – Planet code generator]** Read input from edtPlanet ✓ Get length of input ✓ Initialise an empty string ✓ Loop through each character ✓ Get ASCII value using Ord ✓ *or alternative method* Test if ✓ ASCII value is odd using MOD 2 <> 0 ✓ Build ✓ result string using IntToStr ✓ Add hyphen - only between values (not at start or end) ✓ Display result in lblQ1_3.Caption ✓ Program works correctly with sample input ✓	**12**	
1.4	**BUTTON: [Q1.4 – Habitability check]** Correctly initialise and use a counter ✓ Check each checkbox and increment counter accordingly: cbxAtmosphere ✓ cbxWater ✓ cbxTemperature ✓ cbxMagnetic Field ✓ Correct condition to check if 3 or more are selected ✓ Display correct message in correct component (Habitable) ✓ Else ✓ Display correct message in correct component (Unlikely) ✓	**9**	
	QUESTION 1 TOTAL	**35**	

	QUESTION 2		MAX MARK	MARKS ACHIEVED
2.1	2.1.1	**Button [Q2.1.1]** 'SELECT PlanetType, sum(NumMoons) AS TotalMoons FROM tblPlanets GROUP BY PlanetType ORDER BY sum(NumMoons) DESC' SELECT correct field, ✓ sum(correct field) AS correct alias name ✓ FROM correct table ✓ GROUP BY correct field ✓ ORDER BY correct field ✓ DESC ✓ *ORDER BY either version (with alias name or full expression)	**6**	
	2.1.2	**Button [Q2.1.2]** 'SELECT MissionName, LaunchYear, Success, PlanetName FROM tblPlanets, tblMissions WHERE tblPlanets.PlanetID = tblMissions.PlanetID AND PlanetType = "Gas Giant" AND Success = TRUE' SELECT correct fields ✓ FROM correct tables ✓ WHERE link between tables (tblPlanets.PlanetID ✓ = tblMissions.PlanetID ✓) AND PlanetType = “Gas Giant” ✓ AND Success = TRUE ✓ *Link between tables can also be solved using an INNER JOIN *Boolean TRUE can also be written as -1	**6**	
	2.1.3	**Button [Q2.1.3]** 'INSERT INTO tblMissions (MissionName, LaunchYear, Success, Agency, PlanetID) VALUES ("Karoo Voyager", 2025, TRUE, "SAASA", 5)' INSERT INTO correct table ✓ (correct field names) ✓ VALUES correct data in the correct data type: strings bracketed by “ ”✓ Boolean and integer not bracketed by “ ”✓ *Boolean TRUE can also be written as -1	**4**	

	2.1.4	**Button [Q2.1.4]** 'UPDATE tblMissions SET Agency = "SpaceX", Success = TRUE, LaunchYear = ' + sYear + ' ' + WHERE PlanetID = 8 AND Agency = "NASA" AND Success = FALSE ' Assign current year to sYear variable ✓ * Accept: sYear := Copy(DateToStr(Now), 1, 4) *or* FormatDateTime('yyyy', Now) UPDATE correct table ✓ SET correct fields ✓ Use of Dynamic variable sYear ✓ WHERE correct fields and data ✓ separated by AND ✓ *Boolean FALSE can also be written as 0	**6**	
	2.1.5	**Button [Q2.1.5]** 'SELECT Agency, count(*) AS TotalSuccess FROM tblMissions WHERE Success = TRUE GROUP BY Agency HAVING count(*) >= 3' SELECT correct field, ✓ count(correct field) AS correct alias name ✓ FROM correct table ✓ WHERE correct field = TRUE ✓ GROUP BY correct field ✓ HAVING count(*) >= 3 ✓ *HAVING either version (with alias name or full expression)	**6**	
2.2	2.2.1	**Button [Q2.2.1]** Add the following to the rich edit Number of planets + tblPlanets.RecordCount ✓ Number of missions + tblMissions.RecordCount ✓ *Can be solved with a loop and manually assigned counters	**2**	
	2.2.2	**Button [Q2.2.2]** Retrieve selected planet name from combo box cmbPlanets ✓ Loop through tblPlanets to find a match with selected planet name ✓ Extract PlanetID from the matching record ✓ Loop through tblMissions ✓ using PlanetID and Success = True ✓ Output the planet name as heading using bold formatting ✓ Check if any matching records are found ✓ If not, display message "No successful missions to [PlanetName]" ✓ If yes, loop through missions ✓ Display mission in format: MissionName (Agency, LaunchYear) ✓	**10**	
		QUESTION 2 TOTAL	**40**	

	QUESTION 3		MAX MARK	MARKS ACHIEVED
3.1	3.1.1	**Constructor Create** Constructor heading with correct parameters ✓ correct data type ✓ Assign parameters to attributes ✓✓	**4**	
	3.1.2	**Accessor Method – getDistance** Function heading with String value as return data type ✓ Return the correct message ✓ and conversion of fDistance ✓	**3**	
	3.1.3	**Mutator Method – setMoons** Procedure heading with Integer parameter value ✓ Add the parameter value to fMoons ✓	**2**	
	3.1.4	**Auxiliary Method - calcDensity** Function heading with Real value as return data type ✓ *Suggested solution* Radius = fDiameter ÷ 2 ✓ Volume = (4 ÷ 3) ✓ x Pi ✓ x Power(Radius, 3) ✓ Return ✓ (fMass * Power(10,24)) ✓ ÷ Volume *Any solution that correctly returns the density	**7**	
	3.1.5	**Accessor Method – hasRings** Function heading with Boolean value as return data type ✓ Return fRings ✓	**2**	
	3.1.6	**ToString Method** Function heading with String value as return data type ✓ Build a string with correct labels, attributes, tab spacing ✓ Return the String ✓	**3**	
		Subtotal: Object class	**[21]**	

3.2	3.2.1	**Button [3.2.1 - Instantiate Planet Object]** Extract the planet name from the radio group rgpPlanets. ✓ Assign File ✓ Test to see if the text file exists. ✓ If the text file cannot be found, display a suitable message ✓ and Close the application. ✓ Loop through the text file ✓ ○ Read line ✓ ○ Use string handling to separate data ✓✓ ○ Search for the correct planet. ✓ ○ If the planet is found, extract all the required values for the planet ✓ ○ Instantiate the new Planet object. ○ objPlanet ✓ := TPlanet.Create (correct parameters) ✓ If the planet is not found, display a suitable message. ✓ Display a message to the user that the Planet has been created. Planet instantiated ✓	**15**	
	3.2.2	**Button [3.2.2 – Update number of moons]** Use **setMoons** method with spin edit value to set attribute ✓	**1**	
	3.2.3	**Button [3.2.3 – Display]** Determine if planet has moons using **hasRings** ✓ True = Yes ✓ False = No ✓ *Display the following information in the rich edit* Display object using **toString** method ✓ Display Distance using **getDistance** method ✓ Display correct output of Rings (Yes/No) ✓ Display Density formatted correctly ✓ using **calcDensity** method ✓	**8**	
			[24]	
		QUESTION 3 TOTAL	**45**	

QUESTION 4		MAX MARK	MARKS ACHIEVED
4.1	**Button [4.1 – Display Grid]** Add spacing before heading row (e.g. ' ' or similar) ✓ Create column heading row with numbers 1 to 8 ✓ Add heading to rich edit ✓ Loop through 1 to 8 for each row ✓ Add row number at the start of each line ✓ Loop through each column within the row ✓ Concatenate cell content to the string correctly ✓ Add each row string to the rich edit ✓	**8**	
4.2	**Button [4.2 – Route]** Retrieve the selected column number from sedColumn ✓ Initialise a boolean variable e.g. bClear to True ✓ Loop through each row of the selected column ✓ Correct check for an asterisk ('*') in the column ✓ Set bClear to False if any asterisk is found ✓ Correct conditional logic to determine final message ✓ Output correct message using rich edit ✓ *Alternative Solution* Instead of making use of a Boolean variable, asterisks are counted. If 0, then clear else not clear. Follow same mark allocation as above.	**7**	
4.3	**Button [4.3 – Safest Planet]** Initialise max distance (iMaxDist) to an appropriate starting value ✓ Loop through 1 to 8 for each row ✓ *Outer loop* Loop through each column within the row ✓ *Outer loop* Correctly identify a planet using if ar2Solar[row,col] in ['Y','V','E','M','J','S','U','N'] ✓ Initialise minimum distance (iMinDist) to a large value ✓ Loop through 1 to 8 for each row ✓ *Inner loop* Loop through each column within the row ✓ *Inner loop* Check every cell for asteroids ✓ Correct the calculation of distance using Abs(iRow - r) + Abs(iCol - c) ✓✓ Update iMinDist if current distance is smaller ✓ After checking all asteroids, compare iMinDist with iMaxDist ✓ Update iMaxDist, iBestRow, and iBestCol if this planet is currently the safest ✓ Output planet name using arrPlanets[Pos(...)] logic ✓ Output row, column, and closest asteroid distance to rich edit ✓	**15**	
	QUESTION 4 TOTAL	**30**	

SAMPLE SOLUTIONS

QUESTION 1

////////// 35 Marks //////////

```
// ========================================================
//                    Question 1.1 - 6 Marks
// ========================================================
procedure TfrmQuestion1.btnQ1_1Click(Sender: TObject);
begin
  /// Enter your code below ///
  shpBall := TShape.Create(pnlOutput);
  with shpBall do
    begin
      Parent := pnlOutput;
      Shape  := stCircle;
      Width  := 50;
      Height := 50;
      Brush.Color := clYellow;
      Align       := alClient;
    end;
end;

// ========================================================
//                    Question 1.2 - 8 Marks
// ========================================================
procedure TfrmQuestion1.btnQ1_2Click(Sender: TObject);
var
  rLightYears, rTrillionKM : Real;
  iRounded : Integer;
begin
  /// Enter your code below ///
  rLightYears := StrToFloat(edtDistance.Text);
  rTrillionKM := rLightYears * 9.461;
  iRounded    := Round(rTrillionKM);
  lbl1_2.Caption := FloatToStrF(rLightYears, ffFixed, 6, 0) +
                    ' light years is approximately ' +
                    IntToStr(iRounded) + ' trillion kilometers.';
end;
```



```
// ============================================================
//                        Question 1.3 - 12 Marks
// ============================================================
procedure TfrmQuestion1.btnQ1_3Click(Sender: TObject);
var
  sPlanet, sCode : String;
  K, iASCII : Integer;
begin
  /// Enter your code below ///
  sPlanet := edtPlanet.Text;
  sCode := '';

  for K := 1 to Length(sPlanet) do
  begin
    iASCII := Ord(sPlanet[K]);
    if iASCII mod 2 <> 0 then
    begin
      if sCode <> '' then
        sCode := sCode + '-';
      sCode := sCode + IntToStr(iASCII);
    end;
  end;

  lblQ1_3.Caption := sCode;
end;

// ============================================================
//                        Question 1.4 - 9 Marks
// ============================================================
procedure TfrmQuestion1.btnQ1_4Click(Sender: TObject);
var
  iCount: Integer;
begin
  /// Enter your code below ///
  iCount := 0;

  if cbxAtmosphere.Checked then Inc(iCount);
  if cbxWater.Checked then Inc(iCount);
  if cbxTemperature.Checked then Inc(iCount);
  if cbxMagneticField.Checked then Inc(iCount);

  if iCount >= 3 then
    lblQ1_4.Caption := 'This planet is potentially habitable.'
  else
    lblQ1_4.Caption := 'This planet is unlikely to be habitable.';
end;
```


QUESTION 2

////////// 40 marks //////////

```
// ============================================================
//                    Question 2.1.1 - 6 Marks
// ============================================================
procedure TfrmQuestion2.btn2_1_1Click(Sender: TObject);
// Provided code - DO NOT DELETE OR ALTER //
var
  sSQL1: String;
begin
  /// Enter your code below ///
  sSQL1 := 'SELECT PlanetType, sum(NumMoons) AS TotalMoons ' +
           'FROM tblPlanets ' +
           'GROUP BY PlanetType ' +
           'ORDER BY sum(NumMoons) DESC';

  // Provided code - DO NOT DELETE OR ALTER //
  dbCONN.runSQL(sSQL1);
  if length(sSQL1) <> 0 then
    SetGridColumnWidths(dbgSQL);
end;

// ============================================================
//                    Question 2.1.2 - 6 Marks
// ============================================================
procedure TfrmQuestion2.btn2_1_2Click(Sender: TObject);
// Provided code - DO NOT DELETE OR ALTER //
var
   sSQL2: String;
begin
  /// Enter your code below ///
  sSQL2 := 'SELECT MissionName, LaunchYear, Success, PlanetName ' +
           'FROM tblPlanets, tblMissions ' +
           'WHERE tblPlanets.PlanetID = tblMissions.PlanetID AND ' +
           'PlanetType = "Gas Giant" AND Success = TRUE';

  // Provided code - DO NOT DELETE OR ALTER //
  dbCONN.runSQL(sSQL2);
  if length(sSQL2) <> 0 then
    SetGridColumnWidths(dbgSQL);
end;
```



```
// ============================================================
//                        Question 2.1.3 - 4 Marks
// ============================================================
procedure TfrmQuestion2.btn2_1_3Click(Sender: TObject);
// Provided code - DO NOT DELETE OR ALTER //
var
  sSQL3 : String;
  sLine : String;
begin
  /// Enter your code below ///
  sSQL3 := 'INSERT INTO tblMissions ' +
           '(MissionName, LaunchYear, Success, Agency, PlanetID) ' +
           'VALUES ("Karoo Voyager", 2025, TRUE, "SAASA", 5)';

  // Provided code - DO NOT DELETE OR ALTER //
  dbCONN.executeSQL(sSQL3,dbgPlanets,dbgMissions,dbgSQL);
  if length(sSQL3) <> 0 then
    SetGridColumnWidths(dbgSQL);
end;

// ============================================================
//                        Question 2.1.4 - 6 Marks
// ============================================================
procedure TfrmQuestion2.btn2_1_4Click(Sender: TObject);
// Provided code - DO NOT DELETE OR ALTER //
var
  sSQL4: String;
  sYear: String;
begin
  /// Enter your code below ///
  sYear := copy(DateToStr(now()),1,4);
  sSQL4 := 'UPDATE tblMissions ' +
           'SET Agency = "SpaceX", Success = TRUE, LaunchYear = ' + sYear + ' ' +
           'WHERE PlanetID = 8 AND Agency = "NASA" AND Success = FALSE ' ;

  // Provided code - DO NOT DELETE OR ALTER //
  dbCONN.executeSQL(sSQL4,dbgPlanets,dbgMissions,dbgSQL);
  if length(sSQL4) <> 0 then
    SetGridColumnWidths(dbgSQL);
end;
```



```
// ============================================================
//                        Question 2.1.5 - 6 Marks
// ============================================================
procedure TfrmQuestion2.btn2_1_5Click(Sender: TObject);
// Provided code - DO NOT DELETE OR ALTER //
var
  sSQL5: String;
begin
  /// Enter your code below ///
  sSQL5 := 'SELECT Agency, count(*) AS TotalSuccess ' +
           'FROM tblMissions ' +
           'WHERE Success = TRUE ' +
           'GROUP BY Agency ' +
           'HAVING count(*) >= 3';

  // Provided code - DO NOT DELETE OR ALTER //
  dbCONN.runSQL(sSQL5);
  if length(sSQL5) <> 0 then
    SetGridColumnWidths(dbgSQL);
end;

// ============================================================
//                        Question 2.2.1 - 2 Marks
// ============================================================
procedure TfrmQuestion2.btn2_2_1Click(Sender: TObject);
begin
  // Provided code - DO NOT DELETE OR ALTER //
  with redOutput do
    begin
      Clear;
      SelAttributes.Style := [fsBold];
      Lines.Add('Solar System');
    end;
  /// Enter your code below ///
  redOutput.Lines.add('Number of planets: ' + IntToStr(tblPlanets.RecordCount));
  redOutput.Lines.add('Number of missions: ' + IntToStr(tblMissions.RecordCount));
end;

// ============================================================
//                        Question 2.2.2 - 10 Marks
// ============================================================
procedure TfrmQuestion2.btn2_2_2Click(Sender: TObject);
var
  sPlanetName, sAgency, sMission: String;
  iPlanetID: Integer;
begin
  // Provided code - DO NOT DELETE OR ALTER //
  redOutput.Clear;
  if cmbPlanets.ItemIndex = -1 then
    begin
      dbCONN.ErrorMessage('Please select a planet');
      Exit;
    end;
```

```
  with tblPlanets do
    begin
      Open;
      First;
      /// Enter your code below ///
      sPlanetName := cmbPlanets.Text;

      // Find PlanetID from tblPlanets
      while not Eof do
        begin
          if FieldByName('PlanetName').AsString = sPlanetName then
          begin
            iPlanetID := FieldByName('PlanetID').AsInteger;
            Break;
          end;
          Next;
        end;
    end;

  with tblMissions do
    begin
      // Filter missions to successful missions for that planet
      Filtered := False;
      Filter   := 'PlanetID = ' + IntToStr(iPlanetID) + ' AND Success = True';
      Filtered := True;

      redOutput.Clear;
      redOutput.SelAttributes.Style := [fsBold];
      redOutput.Lines.Add(sPlanetName);
      if RecordCount = 0 then
        redOutput.Lines.Add('No successful missions to ' + sPlanetName)
      else
      begin
        Open;
        First;
        while not Eof do
          begin
            sMission := FieldByName('MissionName').AsString;
            sAgency  := FieldByName('Agency').AsString;
            redOutput.Lines.Add(sMission + ' (' + sAgency + ', ' +
                                FieldByName('LaunchYear').AsString + ')');
            Next;
          end;
      end;

      Filtered := False;
    end;
end;
```


QUESTION 3

////////// 45 Marks //////////

```
// ============================================================
//                        Question 3.1.1 - 4 Marks
// ============================================================
constructor TPlanet.Create(sName, sType: String; rDiameter, rMass, rDistance : Real;
iMoons : Integer; bRings : Boolean);
begin
  fName     := sName;
  fType     := sType;
  fDiameter := rDiameter;
  fMass     := rMass;
  fDistance := rDistance;
  fMoons    := iMoons;
  fRings    := bRings;
end;

// ============================================================
//                        Question 3.1.2 - 3 Marks
// ============================================================
function TPlanet.getDistance: String;
begin
  Result := 'Distance: ' + #9 + FloatToStrF(fDistance, ffFixed, 8, 1) +
            ' million km from the Sun.';
end;

// ============================================================
//                        Question 3.1.3 - 2 Marks
// ============================================================
procedure TPlanet.setMoons(iMoons : Integer);
begin
  fMoons := fMoons + iMoons;
end;

// ============================================================
//                        Question 3.1.4 - 7 Marks
// ============================================================
function TPlanet.calcDensity: Real;
var
  rRadius, rVolume : Real;
begin
  rRadius := fDiameter / 2;
  rVolume := (4 / 3) * Pi * Power(rRadius, 3);
  Result := (fMass * Power(10, 24)) / rVolume;
end;
```



```
// ============================================================
//                        Question 3.1.5 - 2 Marks
// ============================================================
function TPlanet.hasRings: Boolean;
begin
  Result := fRings;
end;

// ============================================================
//                        Question 3.1.6 - 3 Marks
// ============================================================
function TPlanet.toString: String;
begin
  Result := fName + #13#13 +
            'Type:' + #9 + fType + #13 +
            'Diameter:' + #9 + FloatToStr(fDiameter) + #13 +
            'Mass:' + #9 + FloatToStr(fMass) + #13 +
            'Moons:' + #9 + IntToStr(fMoons);
end;
```



```
// ============================================================
//                      Question 3.2.1 - 15 Marks
// ============================================================
procedure TfrmQuestion3.btn3_2_1Click(Sender: TObject);
var
  MyFile : TextFile;
  sLine, sName : String;
  arrFields : array[1..7] of String;
  rDiameter, rMass, rDistance : Real;
  iMoons : Integer;
  bRings, bFound : Boolean;
begin
  /// Enter your code below ///
  sName := rgpPlanets.Items.Strings[rgpPlanets.ItemIndex];

  AssignFile(MyFile, 'Planets.txt');
  try
    Reset(MyFile);
  except
    ShowMessage('FNF');
    Exit;
  end;

  bFound := False;
  while (not eof(MyFile)) AND (bFound = False) do
  begin
    ReadLn(MyFile, sLine);

    arrFields[1] := copy(sLine, 1, Pos(',', sLine)-1); // Name
    delete(sLine, 1, Pos(',', sLine));
    arrFields[2] := copy(sLine, 1, Pos(',', sLine)-1); // Type
    delete(sLine, 1, Pos(',', sLine));
    arrFields[3] := copy(sLine, 1, Pos(',', sLine)-1); // Diameter
    delete(sLine, 1, Pos(',', sLine));
    arrFields[4] := copy(sLine, 1, Pos(',', sLine)-1); // Mass
    delete(sLine, 1, Pos(',', sLine));
    arrFields[5] := copy(sLine, 1, Pos(',', sLine)-1); // Distance
    delete(sLine, 1, Pos(',', sLine));
    arrFields[6] := copy(sLine, 1, Pos(',', sLine)-1); // Moons
    delete(sLine, 1, Pos(',', sLine));
    arrFields[7] := sLine; // Rings

    if UpperCase(arrFields[1]) = UpperCase(sName) then
      begin
        bFound := True;
        rDiameter := StrToFloat(arrFields[3]);
        rMass := StrToFloat(arrFields[4]);
        rDistance := StrToFloat(arrFields[5]);
        iMoons := StrToInt(arrFields[6]);
        bRings := arrFields[7] = 'True';
```

```
        objPlanet := TPlanet.Create(arrFields[1], arrFields[2], rDiameter, rMass, rDistance,
iMoons, bRings);
      end;
  end;

  CloseFile(MyFile);
  if bFound then
    ShowMessage('Planet instantiated')
  else
    ShowMessage('Planet not found');

  // Provided code - DO NOT DELETE OR ALTER //
  btn3_2_2.Enabled := True;
end;

// ============================================================
//                          Question 3.2.2 - 1 Mark
// ============================================================
procedure TfrmQuestion3.btn3_2_2Click(Sender: TObject);
begin
  /// Enter your code below ///
  objPlanet.setMoons(sedMoons.Value);

  // Provided code - DO NOT DELETE OR ALTER //
  btn3_2_3.Enabled := True;
end;

// ============================================================
//                          Question 3.2.3 - 8 Marks
// ============================================================
procedure TfrmQuestion3.btn3_2_3Click(Sender: TObject);
var
  sRings : String;
begin
  /// Enter your code below ///
  sRings := 'No';
  if objPlanet.hasRings then
    sRings := 'Yes';
  redOutput.Lines.add(objPlanet.toString + #13 +
                      objPlanet.getDistance + #13 +
                      'Rings: ' + #9 + sRings + #13 +
                      'Density: ' + #9 + FloatToStrF(objPlanet.calcDensity,ffFixed,25,2));
end;
```


QUESTION 4

////////// 30 Marks //////////

```
// ============================================================
//                      Question 4.1 - 8 Marks
// ============================================================
procedure TfrmQuestion4.btn4_1Click(Sender: TObject);
var
  iRow, iCol : Integer;
  sLine : String;
begin
  // Provided code - DO NOT DELETE OR ALTER //
  redOutput.Clear;

  /// Enter your code below ///
  sLine := '  ';
  for iCol := 1 to 8 do
    sLine := sLine + ' ' + IntToStr(iCol);
  redOutput.Lines.Add(sLine);

  for iRow := 1 to 8 do
    begin
      sLine := IntToStr(iRow) + ' ';
      for iCol := 1 to 8 do
        begin
          sLine := sLine + ' ' + ar2Solar[iRow,iCol];
        end;
      redOutput.Lines.add(sLine);
    end;
end;

// ============================================================
//                      Question 4.2 - 7 Marks
// ============================================================
procedure TfrmQuestion4.btn4_2Click(Sender: TObject);
var
  iRow, iCol : Integer;
  bClear : Boolean;
  sLine : String;
begin
  /// Enter your code below ///
  iCol := sedColumn.Value;
  bClear := True;
  for iRow := 1 to 8 do
    if ar2Solar[iRow,iCol] = '*' then
      bClear := False;

  if bClear then
    sLine := 'Route ' + IntToStr(iCol) + ' is clear.'
  else
    sLine := 'Route ' + IntToStr(iCol) + ' is not clear.';

  redOutput.Lines.Add(#13 + sLine);
end;
```



```
// ============================================================
//                        Question 4.3 - 15 Marks
// ============================================================
procedure TfrmQuestion4.btn4_3Click(Sender: TObject);
var
  iRow, iCol, r, c : Integer;
  iMinDist, iDist, iMaxDist : Integer;
  iBestRow, iBestCol : Integer;
  ch : Char;
begin
  /// Enter your code below ///
  iMaxDist := -1;

  // Loop through grid to find each planet
  for iRow := 1 to 8 do
    for iCol := 1 to 8 do
    begin
      ch := ar2Solar[iRow, iCol];
      if ch in ['Y','V','E','M','J','S','U','N'] then
      begin
        iMinDist := 100;  // Start with high value

        // Check distance to all asteroids
        for r := 1 to 8 do
          for c := 1 to 8 do
            if ar2Solar[r, c] = '*' then
            begin
              iDist := Abs(iRow - r) + Abs(iCol - c);
              if iDist < iMinDist then
                iMinDist := iDist;
            end;

        // If this planet is furthest from any asteroid
        if iMinDist > iMaxDist then
        begin
          iMaxDist := iMinDist;
          iBestRow := iRow;
          iBestCol := iCol;
        end;
      end;
    end;

  // Output result
  redOutput.Lines.Add(#13 + 'The safest planet is ' + arrPlanets[
                      Pos(ar2Solar[iBestRow, iBestCol], 'YVEMJSUN')] +
                      ' at Row ' + IntToStr(iBestRow) +
                      ', Column ' + IntToStr(iBestCol) + '.');
  redOutput.Lines.Add('Closest asteroid is ' + IntToStr(iMaxDist) + ' units away.');
end;
```