# INFORMATION TECHNOLOGY P1

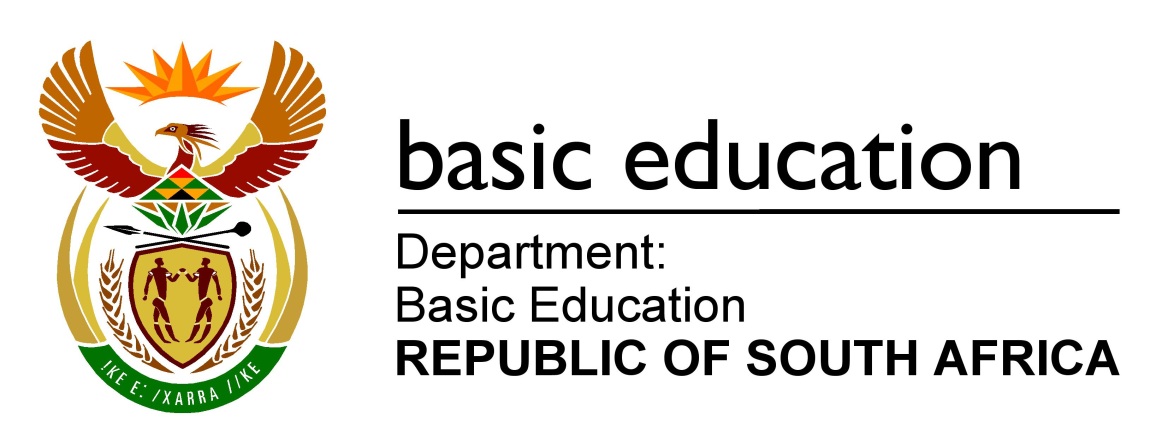
# FEBRUARY/MARCH 2018

# MARKING GUIDELINES

# NATIONAL

# SENIOR CERTIFICATE

# GRADE12



**MARKS: 150**

**These marking guidelines consist of 21 pages.**

|  |  |  |
| --- | --- | --- |
| **GENERAL INFORMATION** |  |  |
|  |  |  |
| * These marking guidelines must be used as the basis for the marking session. They were prepared for use by markers. All markers are required to attend a rigorous standardisation meeting to ensure that the guidelines are consistently interpreted and applied in the marking of candidates' work. |  |  |
|  |  |  |
| * Note that learners who provide an alternate correct solution to that given as example of a solution in the marking guidelines will be given full credit for the relevant solution, unless the specific instructions in the question paper were not followed or the requirements of the question were not met. |  |  |
|  |  |  |
| * **Annexures A, B and C** (pages 3–8) include the marking grid for each question and a table for a summary of the learner’s marks. |  |  |
|  |  |  |
| * **Annexures D, E, and F** (pages 9–21) contain examples of a programming solution for QUESTION 1 to QUESTION 3 in programming code. |  |  |
|  |  |  |
| * Copies of **Annexures A, B and C** (pages 3–8) and the **summary of learner’s marks** (page 8) should be made for each learner and completed during the marking session. |  |  |

**ANNEXURE A**

**SECTION A**

**QUESTION 1: MARKING GRID – GENERAL PROGRAMMING SKILLS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CENTRE NUMBER: | | EXAMINATION NUMBER: | | |
| QUESTION | **DESCRIPTION** | | MAX. MARKS | LEARNER'S MARKS |
|  | ***A learner must be penalised only once if the same error is repeated.*** | |  |  |
| 1.1 | **[Button 1.1 – Total area]**  Extract the radius ✓ convert to real✓  Extract the base and height ✓ from edit boxes  Area of circle = pi \* ✓ Sqr(radius) ✓  Area of triangles = 0.5 \* base \* height ✓ \* number of triangles✓  Display the area of circle✓ with label ✓  Display the area of triangles in total ✓  Calculate total area of circle and triangles ✓  Display the total area to 2 decimal places ✓ | | **12** |  |
| 1.2 | **[Button 1.2 – Next blue moon]**  Find position of colon in label ✓  Extract moonYear from label ✓ and convert to integer ✓  Repeat  Add 3 ✓ to the moonYear ✓  Until moonYear > ✓ year of the current system date ✓  Display moonYear ✓  Correct loop example repeat...until ✓ | | **9** |  |
| 1.3 | **[Button 1.3 – Highest common factor]**  Extract number 1 from edit box ✓ convert to integer✓  Extract number 2 from edit box ✓  Loop x from 1 ✓ to Minimum (number1,number2) ✓  Test if (number mod x = 0) ✓ AND ✓ (number2 mod x =0) ✓  Set hcf to x ✓  Display hcf in the edit box ✓ | | **10** |  |
| 1.4 | **Button [1.4 – Remove vowels]**  Extract sentence from edit box ✓  Set a temp variable to first character of sentence ✓  Loop x from 2 ✓ to length of sentence ✓  if (sent[x - 1] = ' ') ✓ OR ✓  NOT(upcase✓ (sent[x]) ✓ in ['A', 'E', 'I', 'O', 'U']) ✓  temp = temp + sent[x] ✓  Display temp in edit box ✓ | | **11** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 1.5 | **[Button 1.5 – Slide show]**  for initialising total and group number ✓  Set total = 0  Set group number = 0  Use a loop ✓ to test if total < SeatsAvailable(constant given) ✓  Use an input box to enter the number of people ✓  If number of people in group ✓<= (seatsAvailable - total)✓  Increment group number ✓  Increment total by the number of people ✓  Display group number and number of people✓  Else✓  Display message ✓ with correct values ✓  End of loop | **12** |  |
|  | **TOTAL SECTION A:** | **54** |  |

**ANNEXURE B**

**SECTION B**

**QUESTION 2: MARKING GRID - OBJECT-ORIENTED PROGRAMMING**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CENTRE NUMBER: | | EXAMINATION NUMBER: | | |
| QUESTION | **DESCRIPTION** | | MAX. MARKS | LEARNER'S MARKS |
| 2.1.1 | **Constructor:**  Heading with ONLY four values with✓  Three string parameters and one real ✓  Assign parameter values to four attributes✓  Assign false to fNavigationalStatus attribute✓ | | **4** |  |
| 2.1.2 | **accessor METHOD:**  Constellation: return type string✓ returns attribute✓ | | **2** |  |
| 2.1.3 | **setNavigationalStatus** **PROCEDURE:**  Using a procedure✓ (not function)  Receive Boolean value✓  Set value of fNavigationalStatus attribute✓ | | **3** |  |
| 2.1.4 | **determineVisibility FUNCTION:**  If distance less than 80✓  Clearly visible✓  Else✓ {*distance >= 80*}  If distance <= 900 ✓  If magnitude <= 2 ✓  Hardly visible to the naked eye ✓  Else {*magnitude >2*}✓  Visible by means of standard optical aid ✓  Else {*distance > 900*}✓  Only visible by means of specialised optical aid ✓  Set result to visibility value✓ | | **11** |  |
| 2.1.5 | **toString** **METHOD**  Add attributes to output string - name, constellation, magnitude and light years✓ with correct labels ✓  If star is navigational star ✓  Add star name ‘ is a navigational star.’ to message✓  Else  Add star name ‘ is a passive star.’ to message✓  Set result to concatenated string ✓ | | **6** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 2.2.1 | **Button – [2.2.1 - Search holder]:**  Extract the name of the star from the combo box✓  Assign ✓ and reset ✓  Initialise Flag/Counter ✓  Conditional loop (while/repeat) NOT EOF✓ & NOT Found✓  Read line from text file ✓  Compare if line = name of star ✓  Read THREE lines from text file ✓  Instantiate star object - Create objStarX ✓ (param) ✓  Correct number and order of parameters✓  objStarX := TStar.Create(parameters)  Boolean variable set to false to search array✓  Loop from 1 to length of array✓  Test if star name contained in array✓✓  Set Boolean variable to true✓  Call set method for navigational star, ✓ using result of Boolean variable as parameter ✓  Change flag to true ✓  Enable panel pnlButtons ✓  Test if star is NOT found:  Display message ✓  Activate tab sheet 2 ✓  CloseFile✓ | **24** |  |
| 2.2.2 | **Button – [2.2.2 - Display]:**  Using toString method to display object✓  Load constellation picture from file ✓and display✓ | **3** |  |
| 2.2.3 | **Button – [2.2.3 - Visibility]:**  Use the star object to call methods:  objStar.getName✓  objStar.determineVisibility✓  Display in correct format ✓ | **3** |  |
|  | **TOTAL SECTION B:** | **56** |  |

**ANNEXURE C**

**SECTION C**

**QUESTION 3: MARKING GRID – PROBLEM SOLVING PROGRAMMING**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CENTRE NUMBER: | | EXAMINATION NUMBER: | | |
| QUESTION | **DESCRIPTION** | | MAX. MARKS | LEARNER'S MARKS |
| 3.1 | **Button [3.1 – Start game]**  Read level of difficulty from radio group✓  Test for level ✓and assign number of planets to variable✓  Level 1 (50); Level 2 (40); Level 3 (30)  Display 0 on panel; set ItemIndex of combo boxes to 0 ✓  Clear output area for incorrect guesses ✓  Initialise variables for counters✓  *Populate*:  Populate array with - ✓using loops for rows and columns✓  Repeat correct number of times (while/repeat) ✓ to:  Determine random position row ✓and column✓  Test if position does not contain planet ✓ and..  place # in position✓  decrease planets to be placed✓  *Display*:  Clear game board output area✓  Loop through rows✓  Create output string✓  Loop through columns✓  If it is a planet, add #✓ to output string  Else add -✓ to output string  Display output string in game board✓  Enable play button ✓ | | **22** |  |
| 3.2 | **Button [3.2 - Play]**  Accept row and column from combo box✓  Increment number of guesses✓  Find character at the position in array✓  Test if character is a planet✓  Replace with place holder✓  Else✓  Display row ✓and column ✓in area for incorrect guesses  Update display on game board ✓  Display number of guesses and on panel✓  If two planets are found✓  display ‘Won’ message✓  Else  display ’Lost’ message. ✓ | | **13** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 3.3 | **Button [3.3 – Reveal planets]**  Loop through rows ✓  Create output string✓  Loop through columns✓  Add array value to output string✓  Display output string✓ | **5** |  |
|  | **TOTAL SECTION C:** | **40** |  |

**SUMMARY OF LEARNER'S MARKS:**

|  |  |
| --- | --- |
| CENTRE NUMBER: | EXAMINATION NUMBER: |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **SECTION A** | **SECTION B** | **SECTION C** |  |
|  | **QUESTION 1** | **QUESTION 2** | **QUESTION 3** | **GRAND TOTAL** |
| **Max. Marks** | **54** | **56** | **40** | **150** |
| **LEARNER'S MARKS** |  |  |  |  |

**ANNEXURE D: SOLUTION FOR QUESTION 1**

unit Question1\_U;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,

Forms, Dialogs, StdCtrls, ExtCtrls, ComCtrls, jpeg, DateUtils, Math;

type

TfrmQ1 = class(TForm)

PageControl1: TPageControl;

TabSheet1: TTabSheet;

TabSheet2: TTabSheet;

Image1: TImage;

btnQ1\_1: TButton;

redQ1\_1: TRichEdit;

edtRadius: TEdit;

edtBase: TEdit;

edtHeight: TEdit;

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

Label5: TLabel;

Label7: TLabel;

lblInfo: TLabel;

Label9: TLabel;

edtQues1\_2: TEdit;

btnQ1\_2: TButton;

Image2: TImage;

TabSheet4: TTabSheet;

TabSheet5: TTabSheet;

TabSheet6: TTabSheet;

edtSentence: TEdit;

Label10: TLabel;

btnQ1\_4: TButton;

edtRemoveVowels: TEdit;

Label11: TLabel;

Label12: TLabel;

edtNum1: TEdit;

edtNum2: TEdit;

btnQ1\_3: TButton;

edtHCF: TEdit;

redQues1\_5: TRichEdit;

btnQ1\_5: TButton;

procedure btnQ1\_1Click(Sender: TObject);

procedure btnQ1\_2Click(Sender: TObject);

procedure btnQ1\_3Click(Sender: TObject);

procedure btnQ1\_5Click(Sender: TObject);

procedure btnQ1\_4Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

frmQ1: TfrmQ1;

implementation

{$R \*.dfm}

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

// Question 1.1

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

procedure TfrmQ1.btnQ1\_1Click(Sender: TObject);

Const

rPi = 3.14159;

iNumber = 8;

Var

rRadius, rAreaCircle, rBase, rHeight, rAreaTriangles, rTotalArea: real;

begin

rRadius := StrToFloat(edtRadius.Text);

rBase := StrToFloat(edtBase.Text);

rHeight := StrToFloat(edtHeight.Text);

rAreaCircle := rPi \* Sqr(rRadius);

redQ1\_1.Lines.Add('Area of circle = ' + FloatToStr(rAreaCircle));

rAreaTriangles := 0.5 \* rBase \* rHeight \* iNumber;

redQ1\_1.Lines.Add('Total area of triangles = '+

FloatToStr(rAreaTriangles));

rTotalArea := rAreaCircle + rAreaTriangles;

redQ1\_1.Lines.Add('Total area = ' +

FloatToStrF(rTotalArea, ffFixed, 6, 2));end;

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

// Question 1.2

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

procedure TfrmQ1.btnQ1\_2Click(Sender: TObject);

Const

iIncrease = 3;

Var

dDate: TDateTime;

iColon, iMoonYear: integer;

begin

iColon := pos(':',lblInfo.Caption);

iMoonYear := StrToInt(copy(lblInfo.Caption, iColon+1, 4));

repeat

iMoonYear := iMoonYear + iIncrease;

until iMoonYear > YearOf(now);

edtQues1\_2.Text := IntToStr(iMoonYear);

end;

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

// Question 1.3

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

procedure TfrmQ1.btnQ1\_3Click(Sender: TObject);

Var

iNum1, iNum2, iHcf, iLoop, i: integer;

begin

iNum1 := StrToInt(edtNum1.Text);

iNum2 := StrToInt(edtNum2.Text);

for i := 1 to Min(iNum1, iNum2) do

begin

if (iNum1 mod i = 0) and (iNum2 mod i = 0) then

begin

iHcf := i;

end;

end;

edtHCF.Text := IntToStr(iHcf);

end;

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

// Question 1.4

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

procedure TfrmQ1.btnQ1\_4Click(Sender: TObject);

Var

sSent, sTemp: String;

i: integer;

begin

sSent := edtSentence.Text;

sTemp := sSent[1];

for i := 2 to length(sSent) do

begin

if (sSent[i - 1] = ' ') OR

NOT(upcase(sSent[i]) in ['A', 'E', 'I', 'O', 'U']) then

sTemp := sTemp + sSent[i];

end;

edtRemoveVowels.Text := sTemp;

end;

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

// Question 1.5

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

procedure TfrmQ1.btnQ1\_5Click(Sender: TObject);

Const

iSeatsAvailable = 100;

Var

iGroupNumber, iNumPeople, iTotal: integer;

begin

// Provided code

redQues1\_5.Clear;

redQues1\_5.Lines.Add('Group number' + #9#9 + 'Number of people');

iTotal := 0;

iGroupNumber := 0;

while iTotal < iSeatsAvailable do

begin

iNumPeople := StrToInt(Inputbox('',

'Enter the number of people in the group', ''));

if iNumPeople <= (iSeatsAvailable - iTotal) then

begin

Inc(iGroupNumber);

Inc(iTotal,iNumPeople);

redQues1\_5.Lines.Add(IntToStr(iGroupNumber) + #9#9#9 +

IntToStr(iNumPeople));

end //if

else

begin

ShowMessage('Cannot accept a group of ' + IntToStr(iNumPeople) +

' people' + #13 + 'Number of seats available is ' +

IntToStr(100 - iTotal));

end; //else

end;//while

end;

end.

**ANNEXURE E: SOLUTION FOR QUESTION 2**

**OBJECT CLASS:**

unit Star\_U;

interface

type

TStar = class(TObject)

private

// Provided code - attribute declaration

fName: String;

fMagnitude: real;

fDistance: integer;

fConstellation: String;

fNavigationalStatus: Boolean;

public

constructor Create(Name: String; Magnitude: real; Distance: integer;

Constellation: String);

function getConstellation: String;

procedure setNavigationalStatus(bStatus: Boolean);

function determineVisibilty: String;

function toString: String;

// Provided code

function getName: String;

end;

implementation

Uses Math, SysUtils;

{$R+}

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

// Question 2.1.1

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

constructor TStar.Create(sName: String; rMagnitude: real; iDistance: integer; sConstellation: String);

begin

fName := sName;

fMagnitude := rMagnitude;

fDistance := iDistance;

fConstellation := sConstellation;

fNavigationalStatus := false;

end;

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

// Question 2.1.2

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

function TStar.getConstellation: String;

begin

result := fConstellation;

end;

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

// Question 2.1.3

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

procedure TStar.setNavigationalStatus(bStatus: Boolean);

begin

fNavigationalStatus := bStatus;

end;

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

// Question 2.1.4

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

function TStar.determineVisibilty: String;

var

sVisibility: String;

begin

if (fDistance < 80)then

sVisibility := 'Clearly visible'

else

if (fDistance <= 900) then

if (fMagnitude <= 2)then

sVisibility := 'Hardly visible to the naked eye'

else

sVisibility := 'Only visible by means of standard optical aid'

else

sVisibility := 'Only visible by means of specialised optical aid';

Result := sVisibility;

end;

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

// Question 2.1.5

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

function TStar.toString: String;

var

sOutput: String;

begin

sOutput := Format('%s belongs to the' + ' %s constellation.' + #13 +

#13 + 'The star has a magnitude of %3.2f and is %d light years away from Earth.' + #13+ #13,

[fName, fConstellation, fMagnitude, fDistance]);

if fNavigationalStatus then

sOutput := sOutput + fName + ' is a navigational star.'

else

sOutput := sOutput + fName + ' is a passive star.';

result := sOutput;

end;

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

// Provided code

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

function TStar.getName: String;

begin

result := fName;

end;

end.

**MAIN FORM UNIT: QUESTION2\_U.PAS**

unit Question2\_U;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls, Buttons, ComCtrls, ExtCtrls, jpeg, Star\_U;

type

TfrmQ2 = class(TForm)

bmbClose: TBitBtn;

btnQ2\_2\_1: TButton;

redOutput: TRichEdit;

lblHeading: TLabel;

cmbStar: TComboBox;

imgQ2: TImage;

pnlButtons: TPanel;

lblQstNum: TLabel;

btnQ2\_2\_2: TButton;

btnQ2\_2\_3: TButton;

procedure FormCreate(Sender: TObject);

procedure FormCanResize(Sender: TObject; var NewWidth, NewHeight: Integer;

var Resize: Boolean);

procedure cmbStarChange(Sender: TObject);

procedure btnQ2\_2\_1Click(Sender: TObject);

procedure btnQ2\_2\_2Click(Sender: TObject);

procedure btnQ2\_2\_3Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

frmQ2: TfrmQ2;

// Provided code

objStarX: TStar;

implementation

var

arrNavigationStars: array [1 .. 58] of String = (

'Alpheratz','Ankaa','Schedar','Diphda','Achernar','Hamal','Acamar',

'Menkar','Mirfak','Aldebaran','Rigel','Capella','Bellatrix','Elnath',

'Alnilam','Betelgeuse','Canopus','Sirius','Adhara','Procyon','Pollux',

'Avior','Suhail','Miaplacidus','Alphard','Regulus','Dubhe','Denebola',

'Gienah','Acrux','Gacrux','Alioth','Spica','Alkaid','Hadar','Menkent',

'Rigil Kentaurus','Arcturus','Zubenelgenubi','Kochab','Alphecca',

'Antares','Atria','Sabik','Shaula','Rasalhague','Eltanin',

'Kaus Australis','Vega','Nunki','Altair','Peacock','Deneb','Enif',

'Al Na''ir','Fomalhaut','Markab','Polaris');

{$R \*.dfm}

{$R+}

// Question 2.2.1

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

procedure TfrmQ2.btnQ2\_2\_1Click(Sender: TObject);

var

sStarName: String;

tFile: textFile;

sLine, sConstellation: String;

rMagnitude : real;

bFoundFile, bFoundArray: Boolean;

A, iDistance : Integer;

begin

// Question 2.2.1

sStarName := cmbStar.Items[cmbStar.ItemIndex];

AssignFile(tFile, 'StarData.txt');

Reset(tFile);

bFoundFile := false;

While NOT Eof(tFile) AND NOT bFoundFile do

begin

Readln(tFile, sLine);

if Trim(sStarName) = Trim(sLine) then

begin

Readln(tFile, rMagnitude);

Readln(tFile, iDistance);

Readln(tFile, sConstellation);

objStarX := TStar.Create(sStarName, rMagnitude, iDistance,

sConstellation);

bFoundArray := false;

for A := 1 to Length(arrNavigationStars) do

If pos(arrNavigationStars[A], objStarX.getName) > 0 then

bFoundArray := true;

objStarX.setNavigationalStatus(bFoundArray);

bFoundFile := true;

pnlButtons.Show;

end;

end;

if NOT bFoundFile then

begin

sLine := 'The star was not found in the file.';

MessageDlg(sLine, mtError, [mbOK], 0);

pctrlQ2.ActivePageIndex := 1;

pnlButtons.Hide;

end;

CloseFile(tFile);

end;

// Question 2.2.2

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

procedure TfrmQ2.btnQ2\_2\_2Click(Sender: TObject);

begin

// Question 2.2.2

redOutput.Clear; //Provided code

redOutput.Lines.Add(objStarX.toString);

imgQ2.Picture.LoadFromFile(objStarX.getConstellation + '.jpg');

end;

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

// Question 2.2.3

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

procedure TfrmQ2.btnQ2\_2\_3Click(Sender: TObject);

begin

// Question 2.2.3

redOutput.Clear;

redOutput.Paragraph.TabCount := 1;

redOutput.Paragraph.Tab[0] := 50;

redOutput.Lines.Add('Star:' + #9 +objStarX.getName);

redOutput.Lines.Add('Visibility: ' +#9 + objStarX.determineVisibilty);

end;

{$REGION 'Provided code'}

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

// Provided code - DO NOT CHANGE

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

procedure TfrmQ2.cmbStarChange(Sender: TObject);

begin

redOutput.Clear;

pnlButtons.Hide;

imgQ2.Picture := nil;

end;

procedure TfrmQ2.FormCanResize(Sender: TObject;

var NewWidth, NewHeight: Integer; var Resize: Boolean);

begin

Resize := false;

end;

procedure TfrmQ2.FormCreate(Sender: TObject);

begin

CurrencyString := 'R';

pnlButtons.Hide;

end;

{$ENDREGION}

end.

**ANNEXURE F: SOLUTION FOR QUESTION 3**

unit Question3\_U;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, ExtCtrls, ComCtrls, Buttons;

type

TfrmQ3 = class(TForm)

redQ3GameBoard: TRichEdit;

rgbQ3: TRadioGroup;

btnQ3\_1StartGame: TButton;

btnClose: TBitBtn;

btnQ3\_2Play: TButton;

cmbRow: TComboBox;

cmbCol: TComboBox;

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

redQ3Incorrect: TRichEdit;

btnQ3\_3Reveal: TButton;

Label5: TLabel;

pnlQ3NumberOfGuesses: TPanel;

pnlPlay: TPanel;

procedure btnQ3\_1StartGameClick(Sender: TObject);

procedure populate;

procedure display;

procedure btnQ3\_2PlayClick(Sender: TObject);

procedure btnQ3\_3RevealClick(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

frmQ3: TfrmQ3;

iNum: integer;

iFound: integer = 0;

iCount: integer;

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

// Provided code

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

arrGame: array [1..9, 1..9] of char;

implementation

{$R \*.dfm}

{$R+}

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

// Question 3.1

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

procedure TfrmQ3.btnQ3\_1StartGameClick(Sender: TObject);

var

iLevel: integer;

begin

// Question 3.1

iLevel := rgbQ3.ItemIndex;

case iLevel of

0: iNum := 50;

1: iNum := 40;

2: iNum := 30;

end;

iFound := 0;

iCount := 0;

pnlQ3NumberOfGuesses.Caption := IntToStr(iFound);

btnQ3\_2Play.Enabled := true;

redQ3Incorrect.Clear;

cmbRow.ItemIndex:=0;

cmbCol.ItemIndex:=0;

populate;

display;

end;

procedure TfrmQ3.populate;

var

iRow, iCol: integer;

begin

for iRow := 1 to Length(arrGame) do

begin

for iCol := 1 to Length(arrGame) do

begin

arrGame[iRow, iCol] := '-';

end;

end;

while iNum <> 0 do

begin

iRow := random(9) + 1;

iCol := random(9) + 1;

if (arrGame[iRow, iCol] = '-') then

begin

arrGame[iRow, iCol] := '#';

dec(iNum);

end;

end;

end;

procedure TfrmQ3.display;

var

iRow, iCol: integer;

sLine: String;

begin

redQ3GameBoard.Clear;

for iRow := 1 to Length(arrGame) do

begin

sLine := '';

for iCol := 1 to Length(arrGame) do

begin

if arrGame[iRow, iCol] = '$' then

sLine := sLine + '# '

else

sLine := sLine + '- ';

end;

redQ3GameBoard.Lines.Add(sLine);

end;

end;

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

// Question 3.2

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

procedure TfrmQ3.btnQ3\_2PlayClick(Sender: TObject);

var

iRow, iCol: integer;

cChar: char;

begin

// Question 3.2

iRow := StrToInt(cmbRow.text);

iCol := StrToInt(cmbCol.text);

cChar := arrGame[iRow, iCol];

Inc(iCount);

if cChar = '#' then

begin

arrGame[iRow, iCol] := '$';

Inc(iFound);

end

else

begin

redQ3Incorrect.Lines.Add('R' + IntToStr(iRow) + ', C' +

IntToStr(iCol));

end;

display;

pnlQ3NumberOfGuesses.Caption := IntToStr(iCount);

if (iFound >= 2) AND (iCount <= 5) then

begin

btnQ3\_2Play.Enabled := false;

ShowMessage('Game won');

end;

if (iCount >= 5) AND (btnQ3\_2Play.Enabled) then

begin

btnQ3\_2Play.Enabled := false;

ShowMessage('Game lost');

end;

end;

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

// Question 3.3

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

procedure TfrmQ3.btnQ3\_3RevealClick(Sender: TObject);

var

iRow, iCol: integer;

sLine: String;

begin

// Question 3.3

redQ3GameBoard.Clear;

for iRow := 1 to Length(arrGame) do

begin

sLine := '';

for iCol := 1 to Length(arrGame) do

begin

sLine := sLine + arrGame[iRow, iCol] + ' ';

end;

redQ3GameBoard.Lines.Add(sLine);

end;

end;

end.