basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

SENIOR CERTIFICATE EXAMINATIONS

INFORMATION TECHNOLOGY P1

2018

MARKING GUIDELINES

MARKS: 150

These marking guidelines consist of 20 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 3–9) include the marking grid for each question and a table for a summary of the learner’s marks.

- **Annexures D,E and F** (pages 10–20) contain examples of a programming solution for QUESTION 1 to QUESTION 3 in programming code.

- Copies of **Annexures A, B, C** and the **summary of learner’s marks** (pages 3–9) 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	**Form [OnCreate event]** Read the heading from the panel ✓ Uppercase heading✓ + - ✓ + Today's date ✓ Change font style to bold✓ Write heading back to panel✓	**6**	
1.2	**Button [1.2 – Display]** Set output string to empty string✓ Unconditional loop (for) ✓from 'Z' down to 'R'✓ Concatenate✓ cCounter to output string✓ Display output string in output area ✓	**6**	
1.3.1	**Button [1.3.1 – Total number of words]** Declare integer variable for number of pages✓ Read number of pages from spin edit✓ Determine total number of words✓ Write total number of words to edit box ✓as an integer ✓	**5**	
1.3.2	**Radiogroup [1.3.2 – Type of book]** Test if literature selected✓ Divide total number of words✓ by WordsLit variable✓ Else✓ Divide total number of words by WordsTech variable✓ Write total number of words to edit box ✓as an integer ✓	**7**	
1.4	**Combo box [cmbQ1_4]** Declare variables as correct type ✓ Read password from combo box✓ Copy from password, from position 3, ✓ to end✓ //Copy1 Copy from password, from position 1, to position 2✓ //Copy2 Replace password✓ with copy1 + copy2✓ Find position of # in password✓ If position > 0✓ Replace # symbol with $ symbol✓ Insert at position + 1✓ a '.' character✓ Display decrypted password on label✓	**13**	

1.5	**Button [1.5 – Coordinates]** Clear the output area ✓ Display correct heading in output area with open line ✓ Read value for x from edit box✓ and convert to integer ✓ Repeat ✓ (or while with first calculating and displaying first set of coordinates) Calculate y:= m * x ✓ + c ✓ Display coordinates: ✓ format (x,y✓) comma and brackets ✓ Decrease x by 2 ✓ Until y <= 0 ✓ Correct use of a conditional loop ✓	**13**	
	TOTAL SECTION A:	**50**	

ANNEXURE B

SECTION B

QUESTION 2: MARKING GRID – OBJECT ORIENTED PROGRAMMING

CENTRE NUMBER:	**EXAMINATION NUMBER:**		
QUESTION	**DESCRIPTION**	**MAX. MARKS**	**LEARNER'S MARKS**
	A learner must be penalised only once if the same error is repeated.		
2.1.1	Declare attributes • String: (fTitle, fCatNum) ✓✓ • Integer: (fNumBooks, fNumBooksOut) ✓✓	**4**	
2.1.2	**CONSTRUCTOR** • Declaration ✓ • THREE parameters (title, catalogue number, number of books) ✓ with correct data types ✓ • Assignment of parameter values to attributes ✓ • Initialise attribute fNumBooksOut to 0 ✓	**5**	
2.1.3	**METHOD – getCatalogueNumber** • Declaration of method – function ✓ • Result statement ✓	**2**	
2.1.4	**METHOD – getNumBooksOut** • Declaration of method – function ✓ • Result statement ✓	**2**	
2.1.5	**METHOD – setNumBooksOut** • Declaration of method – procedure ✓ with parameter ✓ • Assign number books out to fNumBooksOut attribute ✓	**3**	
2.1.6	**METHOD – isBookAvailable** • Return type – Boolean ✓ • IF statement with correct condition fNumBooks - fNumBooksOut >= 1✓ • Return correct answer (True = if available and False if not available) ✓	**3**	

2.1.7	**METHOD – toString** • Declaration of method – function with return type string ✓ • Concatenate string with attributes ✓ • Correct labels ✓ • Correct conversion of integer values ✓ • Return Result ✓	**5**	
2.2.1	**Button [2.2.1– Instantiate book object]** • Extract the title ✓ and number of copies owned from the edit boxes ✓ • Extract the catalogue number from the combo box ✓ • Instantiate object: *objBook* := *TBook*.Create✓(correct order of parameters✓) • Use **toString** method ✓ to display objects' information ✓ in output area • Make **Update data** button active ✓	**8**	
2.2.2	**Button [2.2.2 – Update data]** • Test IF***BooksData*** text file exists (FileExists()) ✓ • If text file does not exists: o Display a suitable message and Exit event ✓ *IF the **BooksData** file exists:* • AssignFile ✓ Reset ✓ • Initialise books borrowed counter to zero ✓ • Initialise books returned counter to zero ✓ • Use a WHILE Loop through text file ✓ o Read a single line of text from text file ✓ o Extract the catalogue number ✓ using a suitable method ✓ o Test if correct catalogue number ✓ ▪ Extract the action B/R ✓✓ {get last char of string} ▪ Increase the respective counter ✓ using case or if statements ✓ • Calculate books out using counter values ✓ • Call the **setNumBooksOut** method ✓ with the correct argument ✓ • Use **toString** method to display objects' information in output area ✓ • Make the group box **grpProcess** active ✓	**20**	

2.2.3	**Button [2.2.3 – Borrow book]**
	
	Set sCode to '' ✓
	Test if radiogroup itemIndex = 0✓
	    Test if the **isBookAvailable** method✓ returns TRUE ✓
	    o Display a message that the book was borrowed successfully ✓
	    o Set sCode to 'B'✓
	    Else
	        Display a message there are no books available✓
	Else
	    If getNumberBooksOut = 0 ✓
	        Display 'Invalid request' ✓
	    Else
	    o Display a message that the book was returned ✓
	    o Set sCode to 'R' ✓
	
	Test if sCode is not an empty string ✓
	    AssignFile – **BooksData.txt** file ✓
	    Append ✓ to text file
	    Write a line of text✓ in correct format
	    (getCatalogue number#sCode) ✓
	    CloseFile ✓
	Call button btnQ2_2_2✓
		18	

	TOTAL SECTION B	**70**	

ANNEXURE C

SECTION C

QUESTION 3: MARKING GRID – PROBLEM SOLVING

QUESTION	DESCRIPTION	MAX. MARKS	LEARNER'S MARKS
3.1	**Combo box [3.1 – Select name]** Extract staff name from combo box cmbStaff ✓ Display name as part of heading ✓ Loop from 1 to MaxRow✓ Loop from 1 to MaxCol✓ Test if arrPlacements[row,col] ✓ = name✓ Display in output area 'Day' ✓+ day number (column)+ '-' + ✓ 'Library#' ✓+ library number (row) ✓	**10**	

3.2	**Button [3.2 – Create new schedule]**
	
	Input the day library 3 is closed✓
	Loop from 1 to MaxRow ✓
	  Loop from 1 to MaxCol ✓
	    Test if row = 1✓ (or test using case providing for all 3)
	        Assign arrPlacements [r,c] = arrStaff[c] ✓
	    Test if row = 2 ✓
	      Assign arrPlacements [r,c] = arrStaff[(MaxCol + 1)✓ -c]✓
	    Test if row = 3 ✓
	        Test if column value = day closed ✓
	            Set arrPlacements [r,c] = 'XXXXX' ✓
	        Else ✓
	            Repeat ✓
	                Generate a random column from 1 to 6 ✓
	            Until ✓value in
	              array (arrPlacements [row1,col] <> arrStaff[random number] ✓
	              AND ✓
	              array (arrPlacements [row2,col] <> arrStaff[random number])✓
	
	            Set arrPlacements [row,col] ✓= arrStaff[random number] ✓
		20	

	TOTAL	**30**	

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	**50**	**70**	**30**	**150**
LEARNER'S MARKS				

ANNEXURE D: SOLUTION FOR QUESTION 1

```
unit Question1_u;

interface

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

type
  TfrmQ1 = class(TForm)
    pnlQ1_3: TPanel;
    pnlQ1_2: TPanel;
    pnlQ1_1: TPanel;
    btnQ1_2: TButton;
    pnlQ1_5: TPanel;
    pnlQ1_4: TPanel;
    Label1: TLabel;
    Label2: TLabel;
    redQ1_5: TRichEdit;
    Label4: TLabel;
    edtQ1_5: TEdit;
    btnQ1_5: TButton;
    cmbQ1_4: TComboBox;
    label16: TLabel;
    label15: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label5: TLabel;
    spnNumPages: TSpinEdit;
    edtTimeInMinutes: TEdit;
    Label3: TLabel;
    Label9: TLabel;
    edtTotalWords: TEdit;
    GroupBox1: TGroupBox;
    GroupBox2: TGroupBox;
    rgpTypeOfBook: TRadioGroup;
    lblQ1_4: TLabel;
    btnQ1_3_1: TButton;
    edtString: TEdit;
    Label10: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure btnQ1_2Click(Sender: TObject);
    procedure btnQ1_5Click(Sender: TObject);
    procedure cmbQ1_4Change(Sender: TObject);
    procedure rgpTypeOfBookClick(Sender: TObject);
    procedure btnQ1_3_1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmQ1: TfrmQ1;
  // Provided code
  iTotalNumWords: Integer;

implementation

{$R *.dfm}
```

```
// =========================================================================
// Question 1.1
// =========================================================================

procedure TfrmQ1.FormCreate(Sender: TObject);
var
  sHeading: String;
begin
    sHeading := pnlQ1_1.Caption;
    pnlQ1_1.Caption := UpperCase(sHeading) + ' - ' + DateToStr(date);
    pnlQ1_1.Font.style := [fsBold];
end;
// =========================================================================
// Question 1.2
// =========================================================================

procedure TfrmQ1.btnQ1_2Click(Sender: TObject);
var
  // Provided code
  cCounter: Char;
  sOutput: String;
begin
  sOutput := '';
  for cCounter := 'Z' downto 'R' do
  begin
    sOutput := sOutput + cCounter;
  end;
  edtString.Text := sOutput;
end;
// =========================================================================
// Question 1.3.1
// =========================================================================

procedure TfrmQ1.btnQ1_3_1Click(Sender: TObject);
const
  // Provided code
  WordsPerPage = 300;
var
  iNumPages: Integer;
begin
  iNumPages := spnNumPages.Value;
  iTotalNumWords := WordsPerPage * iNumPages;
  edtTotalWords.Text := IntToStr(iTotalNumWords);
end;
// =========================================================================
// Question 1.3.2
// =========================================================================

procedure TfrmQ1.rgpTypeOfBookClick(Sender: TObject);
const
  // Provided code
  WordsTech = 75;
  WordsLiterature = 250;
var
  iTotalMinutes: Integer;
begin
  if rgpTypeOfBook.ItemIndex = 0 then
  begin
    iTotalMinutes := iTotalNumWords DIV WordsLiterature;
  end
  else
  begin
    iTotalMinutes := iTotalNumWords DIV WordsTech;
  end;
  edtTimeInMinutes.Text := IntToStr(iTotalMinutes);
end;
// ============================================================================
// Question 1.4
// ============================================================================

procedure TfrmQ1.cmbQ1_4Change(Sender: TObject);
var
  sPassword: String;
  iPos: Integer;
begin
  sPassword := cmbQ1_4.Text;
  sPassword := copy(sPassword, 3) + copy(sPassword, 1, 2);
  iPos := pos('#', sPassword);
  if iPos > 0 then
  begin
    sPassword[iPos] := '$';
    insert('.', sPassword, iPos + 1);
    // Alternative solution
    // sPassword := copy(sPassword,1,iPos-1) + '$.' + copy(sPassword,iPos+1);
  end;
  lblQ1_4.Caption := sPassword;
end;

// ============================================================================
// Question 1.5
// ============================================================================
procedure TfrmQ1.btnQ1_5Click(Sender: TObject);
var
  iX, iY: Integer;
  iM, iC: Integer;
begin
  // Provided code
  iM := 3;
  iC := -2;

  redQ1_5.Clear;
  redQ1_5.Lines.Add('Coordinates' + #13);
  iX := StrToInt(edtQ1_5.Text);
  repeat
    iY := iM * iX + iC;
    redQ1_5.Lines.Add('(x,y) = (' + IntToStr(iX) + ',' + IntToStr(iY) + ')');
    iX := iX - 2;
  until iY <= 0;
   //Conditional Loop
end;

end.
```

ANNEXURE E: SOLUTION FOR QUESTION 2

OBJECT CLASS:

```
unit Book_U;

interface

uses SysUtils, Dialogs, Math;

// ============================================================================
// Question 2.1.1
// ============================================================================
type
   TBook = class(TObject)
     private
       fTitle        : String;
       fCatNum       : String;
       fNumBooks     : Integer;
       fNumBooksOut  : Integer;

     public
       constructor Create(sBookTitle, sCatelogueNumber : String;
                          iNumBooks : Integer);
       function getCatalogueNumber : String;
       function getNumBooksOut: Integer;
       procedure setNumBooksOut(iNumBooksOut: Integer);
       function isBookAvailable : Boolean;
       function toString : String;
   end;

implementation

{ TBook }

// ============================================================================
// Question 2.1.2
// ============================================================================
constructor TBook.Create(sBookTitle, sCatelogueNumber : String;
                         iNumBooks: Integer);
begin
  fTitle            := sBookTitle;
  fCatNum           := sCatelogueNumber;
  fNumBooks         := iNumBooks;
  fNumBooksOut      := 0;
end;

// ============================================================================
// Question 2.1.3
// ============================================================================
function TBook.getCatalogueNumber: String;
begin
  Result := fCatNum;
end;

// ============================================================================
// Question 2.1.4
// ============================================================================
function TBook.getNumBooksOut: Integer;
begin
  Result := fNumBooksOut;
end;
```

```
// ============================================================================
// Question 2.1.5
// ============================================================================
procedure TBook.setNumBooksOut(iNumBooksOut: Integer);
begin
   fNumBooksOut := iNumBooksOut;
end;

// ============================================================================
// Question 2.1.6
// ============================================================================
function TBook.isBookAvailable: Boolean;
begin
   Result := fNumBooks - fNumBooksOut >= 1;
//Alternative:
//   if fNumAvailable >= 1 then
//     Result := True
//   else
//     Result := False;

// OR if fNumAvailable < 1 then
//     Result := False
//   else
//     Result := True;
end;

// ============================================================================
// Question 2.1.7
// ============================================================================
function TBook.toString: String;
var
  sMsg : String;
begin
  sMsg := 'The library owns %d copies of the book titled %s [%s].' + #13 +
           #13 + 'The number of copies currently out is %d.';
  Result := Format(sMsg, [fNumBooks, fTitle, fCatNum,  fNumBooksOut]);
end;

end.
```

MAIN FORM UNIT: QUESTION2_U.PAS

```
unit Question2_U;

interface

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

type
  TQuestion2 = class(TForm)
    grpBInfo: TGroupBox;
    lblT: TLabel;
    edtTitle: TEdit;
    lblC: TLabel;
    lblS: TLabel;
    edtNumStock: TEdit;
    pnlButn: TPanel;
    bmbclose: TBitBtn;
    redQ2: TRichEdit;
    cmbCatNum: TComboBox;
    btnQ2_2_1: TButton;
    btnQ2_2_2: TButton;
    btnQ2_2_3: TButton;
    grpProcess: TGroupBox;
    rgpProcess: TRadioGroup;
    procedure btnRestoreClick(Sender: TObject);
    procedure btnQ2_2_1Click(Sender: TObject);
    procedure btnQ2_2_2Click(Sender: TObject);
    procedure btnQ2_2_3Click(Sender: TObject);
    procedure cmbCatNumChange(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Question2: TQuestion2;

implementation
// Provided code
uses
  Book_U;

{$R *.dfm}
{$R+}

var
   objBook : TBook;
   arrTitles :array [1..4] of String =
      ('Basic Mathematics', 'Lateral Thinking', 'Coding in Objects',
        'Marketing Magic');
   arrNumberOwn :array [1..4] of Integer =(4, 3, 3, 5);
```

```
// ============================================================================
// Question 2.2.1
// ============================================================================
procedure TQuestion2.btnQ2_2_1Click(Sender: TObject);
var
  sTitle, sCatNum : String;
  iNumStock : Integer;
begin
  sCatNum   := cmbCatNum.Items[cmbCatNum.ItemIndex + 1];
  sTitle := edtTitle.Text;
  iNumStock := StrToInt(edtNumStock.Text);
  objBook := TBook.Create(sTitle, sCatNum, iNumStock);
  redQ2.Clear;
  redQ2.Lines.Add(objBook.toString);
  btnQ2_2_2.Enabled := true;
end;

// ============================================================================
// Question 2.2.2
// ============================================================================
procedure TQuestion2.btnQ2_2_2Click(Sender: TObject);
var
 TFile : TextFile;
 sLine, sCatNum : String;
 cAction        : Char;
 iCountBorrowed, iCountReturned, iOut : Integer;
begin
  redQ2.Clear;
  if NOT FileExists('BooksData.txt') then
   begin
     MessageDlg('File not found.', mtError, [mbOK], 0);
     Exit;
   end;
  redQ2.Clear;
  AssignFile(TFile, 'BooksData.txt');
  Reset(TFile);
  iCountBorrowed := 0;
  iCountReturned := 0;
  while not Eof(TFile) do
    begin
      Readln(TFile, sLine);
      sCatNum := Copy(sLine, 1, Pos('#', sLine)-1);
      if objBook.getCatalogueNumber = sCatNum then
       begin
         cAction := sLine[Length(sLine)];
         case cAction of                   // Or use 2 if statements
          'B' : inc(iCountBorrowed, 1);    // Or use if .. then .. else
          'R' : inc(iCountReturned, 1);
         end;
       end; //if
    end;//while
  CloseFile(TFile);
  iOut := iCountBorrowed - iCountReturned;
  objBook.setNumBooksOut(iOut);
  redQ2.Lines.Add(objBook.toString);
  grpProcess.Enabled := true;
end;
```

```
// ============================================================================
// Question 2.2.3
// ============================================================================
procedure TQuestion2.btnQ2_2_3Click(Sender: TObject);
 var
   sCatNum, sCode : String;
   TFile : TextFile;
begin
   sCode := '';
   if rgpProcess.ItemIndex = 0 then
     begin
         if objBook.isBookAvailable then
          begin
            ShowMessage('The book has been borrowed successfully.');
            sCode := 'B';
          end
       else
          ShowMessage('No copies available');
      end
  else
    if objBook.getNumBooksOut = 0 then
       ShowMessage('Invalid request')
    else
      begin
        sCode := 'R';
        ShowMessage('The book has been returned successfully.');
      end;

   if sCode > '' then
    begin
     AssignFile(TFile, 'BooksData.txt');
     Append(TFile);
     Writeln(TFile,objBook.getCatalogueNumber + '#'+ sCode);
     CloseFile(TFile);
     rgpProcess.ItemIndex := -1;
    end;
    btnQ2_2_2.click;
end;

// Provided code
procedure TQuestion2.btnRestoreClick(Sender: TObject);
var
   e: Boolean;
begin
   DeleteFile('BooksData.txt');
   CopyFile('BooksDataBackup.txt','BooksData.txt', e);
end;


// Provided code
procedure TQuestion2.cmbCatNumChange(Sender: TObject);
var
  iIndex : Integer;
begin
  iIndex :=  cmbCatNum.ItemIndex + 1;
  edtNumStock.Text := IntToStr(arrNumberOwn[iIndex]);
  edtTitle.Text := arrTitles[iIndex];
end;

end.
```

ANNEXURE F: SOLUTION FOR QUESTION 3

```
unit Question3_U;

interface

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

type
  TfrmQ3 = class(TForm)
    redOutput: TRichEdit;
    GroupBox1: TGroupBox;
    btnQ3_1: TButton;
    GroupBox2: TGroupBox;
    cmbStaff: TComboBox;
    Panel1: TPanel;
    Image1: TImage;
    procedure btnQ3_1Click(Sender: TObject);
    procedure Display;
    procedure cmbStaffChange(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

// ============================================================================
// Provided code: Declarations
// ============================================================================
const
  iMaxRow = 3;
  iMaxCol = 6;

var
  frmQ3: TfrmQ3;
  arrStaff: array [1 .. 6] of String = (
    'Trevor',
    'Nkosi',
    'Tamzin',
    'Anette',
    'Bongi',
    'Simon'
  );

  arrPlacements: array [1 .. 3, 1 .. 6] of String =
    (('Nkosi', 'Simon','Anette', 'Bongi', 'Tamzin', 'Trevor'),
     ('Anette', 'Tamzin','Simon', 'Trevor', 'Bongi', 'Nkosi'),
     ('Bongi', 'XXXXX', 'Trevor','Nkosi', 'Nkosi', 'Tamzin'));

implementation

{$R *.dfm}
```

```
// ============================================================================
// Provided code: Procedure Display
// ============================================================================
procedure TfrmQ3.Display;
Var
  sLine: String;
  iCnt, iR, iC: integer;
begin
  redOutput.Clear;
  sLine := '' + #9#9;
  for iCnt := 1 to iMaxCol do
  begin
    sLine := sLine + 'Day ' + IntToStr(iCnt) + #9#9;
  end;
  redOutput.Lines.Add(sLine);

  for iR := 1 to iMaxRow do
  begin
    sLine := 'Library ' + IntToStr(iR) + #9;
    for iC := 1 to iMaxCol do
    begin
      sLine := sLine + arrPlacements[iR, iC] + #9#9;
    end;
    redOutput.Lines.Add(sLine);
  end;
end;

// ============================================================================
// Provided code: FormActivate
// ============================================================================

procedure TfrmQ3.FormActivate(Sender: TObject);
Begin
  Display;
end;

// =============================================================================
// Question 3.1
// =============================================================================
procedure TfrmQ3.cmbStaffChange(Sender: TObject);
Var
 iRow, iColumn: Integer;
  sName: String;
begin

  sName := cmbStaff.Text;
  redOutput.Lines.Add(#13 + sName + '''s  ' + 'Schedule');
  for iColumn := 1 to iMaxCol do
  begin
    for iRow := 1 to iMaxRow do
    begin
      if arrPlacements[iRow, iColumn] = sName then
        redOutput.Lines.Add('Day ' + IntToStr(iColumn) + '-Library#' + IntToStr
            (iRow));
    end;
  end;
end;
```

```
// ============================================================================
// Question 3.2
// ============================================================================
procedure TfrmQ3.btnQ3_2Click(Sender: TObject);
Var
   iRow, iColumn, iNum, iDayClosed: Integer;
begin
  iDayClosed := StrToInt(InputBox('Library 3 closed',
      'Enter the day number when library 3 is closed', '1'));
  for iRow := 1 to iMaxRow do
  begin
    for iColumn := 1 to iMaxCol do
    begin

      case iRow of
        1:
          arrPlacements[iRow, iColumn] := arrStaff[iColumn];
        2:
          arrPlacements[iRow, iColumn] := arrStaff[7 - iColumn];
        3:
          if iColumn = iDayClosed then
          begin
            arrPlacements[iRow, iColumn] := 'XXXXX';
          end
          else
          begin
            repeat
              iNum := Random(6) + 1;
            until ((arrPlacements[1, iColumn] <> arrStaff[iNum])) AND
              ((arrPlacements[2, iColumn] <> arrStaff[iNum]));
            arrPlacements[iRow, iColumn] := arrStaff[iNum];
          end; // case
      end; // else
    end; // iColumn
  end; // iRow

  // Code provided
  Display;

end;

end.
```