KZN COMMON PAPER: JUNE 2018

INFORMATION TECHNOLOGY PAPER 1

MARKING MEMORANDUM / MARKING RUBRIC / SAMPLE SOLUTION

QUESTION 1

QN	DESCRIPTION	MAX	MARK
1.1	Caption set correctly✓ Tests RadioGroup first item selected: ✓ Panel Colour set to White✓ Tests Radio Group 2nd Item selected✓ Font set to Italic✓	5	
1.2	Extracts selected name from List Box✓ Extracts transaction total✓ Evaluates Checkbox and if selected✓, calculates tip amount✓ Outputs "Details of Transaction on" ✓ Combined with System Date ✓ And System Time✓ Outputs Waiter name with caption✓ Outputs Transaction amount with caption✓ Outputs Tip amount with caption✓	10	
1.3	Extracts name from Edit Box✓ Uses flag variable (or similar structure) to track validity of name✓ Loops through name; character by character checking for validity✓ If invalid✓✓, flag (or similar structure) changed accordingly. ✓ If name is invalid, displays appropriate message✓ If name is valid: ✓ Check if Length of Name is divisible by 2: ✓ Extracts first 3 characters from name✓ Swops character 1 and 3 (or reverses entire string) ✓✓✓ ELSE✓ Correctly extracts and combines character at: ✓ Position 1✓ Position (Ceil(Length / 2)) ✓✓ // Some learners may use TRUNC or DIV Position (Length) ✓ Displays generated code correctly✓	20	
1.4	Extract data from sedStart and sedEnd ✓ File assigned and opened (Reset) ✓ Total and Counter assigned to 0✓ Loop structure✓ ReadLn statement correctly written✓ Caters for Start✓ and End position✓; only adding✓/counting ✓the lines within that range. Average calculated correctly✓ VAT calculated correctly✓ Outputs total correctly✓ Outputs Number of Transactions correctly✓ Outputs Average correctly✓ Outputs VAT correctly✓	15	
	TOTAL:	**50**	

SAMPLE SOLUTION:

```
procedure TForm1.rgpQ1_1Click(Sender: TObject);
begin
 {Question 1.1}
 pnlMessage.Caption := 'Welcome to Leblanc'; ✓

 if rgpQ1_1.ItemIndex = 0 then✓
 begin
   pnlMessage.Color := clWhite; ✓
 end
 else✓
 begin
   pnlMessage.Font.Style := [fsItalic]; ✓
 end;

end;
```

[5]

```
procedure TForm1.btn1_2Click(Sender: TObject);
var
  rAmt, rTip : Real;
  sName : String;
begin
 {Question 1.2}
 sName := lstWaiters.Items[lstWaiters.ItemIndex]; ✓
 rAmt := StrToFloat(edtTrans.Text); ✓

 if chbTip.Checked then✓
   rTip := rAmt * 0.1  ✓
 else
   rTip := 0;

 redQ1_2.Text := 'DETAILS OF TRANSACTION ON ' ✓+ DateToStr(Now) ✓+ ' AT ' + TimeToStr(Now) ✓;
 redQ1_2.Lines.Add('');

 redQ1_2.Lines.Add('Waiter: ' + sName); ✓
 redQ1_2.Lines.Add('Transaction Amount: '+FormatFloat('R###0.00', rAmt)); ✓
 redQ1_2.Lines.Add('Tip: '+FormatFloat('R####0.00', rTip)); ✓

end;
```

[10]

```
procedure TForm1.btnValidate1_3Click(Sender: TObject);
var
 sName, sCode : String;
 bValidName : Boolean;
 i : Integer;
 cTemp : Char;
```

```
begin
 {Question 1.3.1}
 sName := edtEntrantName.Text; ✓

 bValidName := TRUE; ✓

 for i := 1 to Length(sName) do✓
 begin
  if not (sName[i] in ['A'..'Z', 'a'..'z'✓]) then✓
  begin
    bValidName := FALSE; ✓
  end;
 end;

 if not bValidName then        }
 begin                         }
   showMessage('Invalid Name'); }  ✓
 end
 else✓
 begin
  sCode := '';

  if Length(sName) mod 2 = 0 then ✓
  begin
   sCode := copy(sName, 1, 3); ✓ // Some learners may copy the characters in reverse

   cTemp := sCode[1]; ✓
   sCode[1] := sCode[3]; ✓
   sCode[3] := cTemp; ✓

  end
  else ✓
  begin
   i := Ceil(Length(sName) / 2); ✓
   sCode ✓:= sName[1] ✓ + sName[i] ✓+ sName[Length(sName)] ✓;
  end;

  edtGenCode1_3.Text := sCode; ✓
 end;
end;
```

[20]

```
procedure TForm1.btnProcess1_4Click(Sender: TObject);
var
 iStart, iEnd, i : Integer;
 t : TextFile;
 sLine : String;
 rTotal, rNum, rAve, rVAT : Real;
```

```
begin
 {Question 1.4}
 iStart := sedStart.Value;  ] ✓
 iEnd := sedEnd.Value;      ]

 AssignFile(t, 'sales.txt'); ]    ✓
 Reset(t);                   ]

 rTotal := 0; ✓
 rNum := 0;

 for i := 1 to 20 do✓
 begin
  ReadLn(t, sLine); ✓
  if (i >= iStart) ✓AND (i <= iEnd) then✓
  begin
    rTotal := rTotal + StrToInt(sLine); ✓
    rNum := rNum + 1; ✓
  end;
 end;

 rAve := rTotal / rNum; ✓

 rVAT := rTotal * 0.15; ✓

 edtTot1_4.Text := FormatFloat('R####.00', rTotal); ✓
 edtNumTrans1_4.Text := FloatToStr(rNum); ✓
 edtAveSpend1_4.Text := FormatFloat('R####.00', rAve); ✓
 edtVAT1_4.Text := FormatFloat('R####.00', rVAT); ✓

end;
```

[15]

SUB-TOTAL: 50

QUESTION 2

QN	DESCRIPTION	MAX	MARK
2.1	Correctly declares: fUserCode✓ fDataUsed✓ fFUPActive✓ All attributes are of the correct data type✓ Correctly declares procedure checkFUP in Interface of unit✓ genUserCode Generates 3 random numbers in correct range ✓✓✓ Combines generated numbers with '-' placed correctly ✓✓✓ setDataUsed Attribute assigned to formal parameter correctly✓ CheckDataUsed Evaluates if data used is less than 500✓ Returns attribute with MB as unit✓ Evaluates if data < 1025 correctly✓ Converts attribute to GB correctly✓ and returns value with GB as unit✓ Evaluates if data usage has exceeded 1024✓ Returns the word "Capped" ✓ CheckFUP Assigns fFUPActive✓ to TRUE or FALSE✓ based on fDataUsed value✓. toString Caption and UserCode output correctly✓ Caption and CheckDataUsed method called correctly✓ Use of tab spaces and new lines used correctly✓	25	
2.2	Instantiation using Default Constructor correct✓ genUserCode called correctly✓ setDataUsed correctly called and actual parameter sent correctly✓ checkFUP called correctly✓ toString called correctly and assigned to redOut✓	5	
	TOTAL	30	

SAMPLE SOLUTION:

2.1

```
private

  fUserCode : String; ✓
  fDataUsed : Integer; ✓          ✓
  fFUPActive : Boolean; ✓

public

procedure genUserCode;

procedure setDataUsed(iData : Integer);

procedure checkFUP; ✓

function CalcDataUsed : String;

function toString : String;


  end;

implementation
uses SysUtils, Math;

{ TClient }

procedure TClient.genUserCode;
var
   iNum : Integer;
begin
  iNum := RandomRange✓ (100✓, 1000 ✓);
  fUserCode := IntToStr(iNum) + '-'; ✓
  iNum := RandomRange(100, 1000);
  fUserCode := fUserCode + IntToStr(iNum) + '-';     ✓
  iNum := RandomRange(100, 1000);
  fUserCode := fUserCode + IntToStr(iNum); ✓
end;


procedure TClient.setDataUsed(iData: Integer);
begin
  fDataUsed := iData; ✓
end;

function TClient.CalcDataUsed: String;
begin
```

```
  if fDataUsed < 500 then ✓
  begin
    Result := IntToStr(fDataUsed) + ' MB'; ✓
  end
  else if fDataUsed < 1025 then ✓
  begin
    Result := FormatFloat('#0.00', fDataUsed / 1024✓) + ' GB'; ✓
  end
  else ✓
  begin
   Result := 'Capped'; ✓
  end;
end;
```

procedure TClient.checkFUP;

```
begin
   fFUPActive := ✓ (FDataUsed >✓ 1024); ✓
end;
```

function TClient.toString: String;

```
begin
  Result := 'User Code:' + #9 + fUserCode + #13 +✓
          'Usage:' + #9 + CheckDataUsed; ✓                    ✓
end;

end.
```

[25]

2.2

procedure TForm1.btnProcessClick(Sender: TObject);

```
begin
  // Question 2.2

  objClient := TClient.Create; ✓
  objClient.genUserCode; ✓
  objClient.setDataUsed(sedDataUsed.Value); ✓
  objClient.checkFUP; ✓
  redOut.Text := objClient.toString; ✓
end;
```

[5]

SUB-TOTAL: 30

QUESTION 3

QN	DESCRIPTION	MAX	MARK
	ALL OPERATIONS CAN BE APPLIED TO EITHER THE ADOTable (tblStaff) or DataSource (dsStaff). This rubric makes reference to the ADOTable, however full credit should be given to learners who make use of the DataSource instead.		
3.1.1	Iterator moved to position 1 of table✓ Sum variable assigned to 0✓ Loop through table✓ Sum incremented✓ by value extracted from table✓ Table iterator moved to next position✓ Average calculated correctly✓✓ Average displayed ✓correctly formatted as currency ✓	12	
	MaleCount assigned to 0 FemaleCount assigned to 0 Move to position 1 in table Loop through table Test correct field from table if Male Increment MaleCount Test correct field from table if Female Increment FemaleCount Move to next record in table Display Male:Female ratio using appropriate statement	12	
3.2.1	Select * ✓from tblStaff✓ ORDER BY✓ Earnings DESC✓	4	
3.2.2	SELECT✓ Waiter, ✓ FORMAT✓ (Earnings * 0.1✓, “Currency”) AS Tax✓ FROM ✓ tblStaff✓	7	
	TOTAL	35	

SAMPLE SOLUTION:

```
procedure TForm1.btnCalcAveClick(Sender: TObject);
var
 rAve : Real;
begin
 // 3.1.1 Calculate Average using Code Construct
 tblStaff.First; ✓
 rAve := 0; ✓
 while  not  ✓tblStaff.Eof do✓
 begin
  rAve := rAve ✓+ tblStaff['Earnings']; ✓
  tblStaff.Next; ✓
 end;

 rAve := rAve / ✓tblStaff.RecordCount; ✓
 showMessage('Average Earnings: ' +✓ FormatFloat ('R######.00'✓, rAve)); ✓

end;

procedure TForm1.btnCountGenClick(Sender: TObject);
var
 iMale, iFemale : Integer;
begin
 // Write your code for Q3.1.2 here

 iMale := 0; ✓
 iFemale := 0; ✓

 tblStaff.First; ✓

 while not tblStaff.EOF do✓
 begin

  if tblStaff['Gender'] = 'M' then✓
  begin
   inc(iMale); ✓
  end
  else ✓
  begin
    inc(iFemale); ✓
  end;

  tblStaff.Next; ✓
 end;

 showMessage('M:F Ratio is '✓ + IntToStr(iMale) + ':'✓ + IntToStr(iFemale)); ✓
end;
```

[24]

3.2.1 Select * ✓from tblStaff✓ ORDER BY✓ Earnings DESC✓ (4)

3.2.2 SELECT✓ Waiter,✓ FORMAT✓ (Earnings * 0.1✓, “Currency”) AS Tax✓ FROM ✓ tblStaff✓ (7)

[11]

SUB-TOTAL: 35

QUESTION 4

QN	DESCRIPTION	MAX	MARK
4.1	Row loop correct✓ Displays correctly called value from arrTypes and leaves a tabspace✓ Column Loop correct✓ Checks if sales under 50✓ Displays "Bad" ✓ Checks if sales > 50 and <100 ✓ Displays "Ave" ✓ Checks if sales > 99✓ Display "Good" ✓ Tabspaces for each tier of output ✓ Evaluation of current row/column correctly in all 3 if tests ✓ Moves to new line✓	12	
4.2	Declares array / list / similar structure to store Average sales ✓ DETERMINE AVERAGES Row Loop✓ Sum variable assigned to 0✓ Column Loop✓ Sum variable incremented✓ by value from Grid✓ Average calculated ✓and assigned to Array / List / Variable✓ SORT Outer loop ✓ Use of Flag variable / counter variable ✓ Inner loop ✓ Compares two items ✓✓✓ Swops items in list / array / variables ✓✓✓ Swops items in parallel structure arrTypes ✓ DISPLAY Loop✓ Displays from arrTypes ✓ TabSpace✓ Converted value✓ from List/Array/Variable✓	23	
	TOTAL	35	

SAMPLE SOLUTION:

```
procedure TForm1.btnSalesProcess4_1Click(Sender: TObject);
var
 r, c : Integer;
begin
  for r := 1 to 5 do ✓
  begin
    redOut.SelText := arrTypes[r]  + #9; ✓
    for c := 1 to 4 do✓
    begin
      if arrSales[r][c] < 50 then ✓
        redOut.SelText := 'Bad'✓ + #9
      else if arrSales[r][c] < 100 then✓
        redOut.SelText := 'Ave'✓ + #9
      else✓
        redOut.SelText := 'Good'✓ + #9;
    end;
    redOut.SelText := #13; ✓
  end;
end;
```

✓ - tabspaces
✓ - Row/Column index used correctly

[12]

```
procedure TForm1.btnGenReport4_2Click(Sender: TObject);
var
 arrTot : Array[1..5] of Real; ✓
 r, c : Integer;
 rTemp : Real;
 bSwop : Boolean;
 sTemp : String;
begin
 for r := 1 to 5 do ✓
 begin
  arrTot[r] := 0; ✓
  for c := 1 to 4 do✓
  begin
    arrTot[r] := arrTot[r]  +✓ arrSales[r][c]; ✓
  end;
  arrTot[r] := arrTot[r] ✓ / 4; ✓
 end;

 repeat
   bSwop := FALSE;
   for c := 1 to 4 do✓
   begin
     if✓ arrTot[c] ✓< arrTot[c + 1] ✓then
     begin
       rTemp := arrTot[c]; ✓
       arrTot[c] := arrTot[c + 1]; ✓
       arrTot[c + 1] := rTemp; ✓
```

```
        sTemp := arrTypes[c];              }
        arrTypes[c] := arrTypes[c + 1];    }  ✓
        arrTypes[c+1] := sTemp;            }
        bSwop := TRUE; ✓
      end;

    end;
  until bSwop = FALSE; ✓

  for r := 1 to 5 do✓
  begin
    redReport.Lines.Add(arrTypes[r] ✓ + #9 ✓ + FloatToStr✓ (arrTot[r])); ✓
  end;

end;
```

[23]

SUB-TOTAL: 35

GRAND TOTAL: 150