**IT P1 – KZN COMMON PAPER Trials 2017 : Marksheet**

**Name\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Question 1** | **Question 2.1** | **Question 2.2** | **Question 3** | **Total** |
| **/ 50** | **/ 24** | **/ 31** | **/45** | **/150** |

**Question 1: Possible Solution**

|  |  |  |
| --- | --- | --- |
| Aspect | Mark | Learner |
| **Question 1.1**  procedure TfrmQ1.btnRegisterClick(Sender: TObject);  var  I: Integer;  begin  // question 1.1  if Length(edtCell.Text) <> 10 then✓  begin  edtCell.Clear; ✓  edtCell.SetFocus; ✓  ShowMessage('Cell number must 10 digits long'); ✓  exit;  end  else  begin  for I := 1 to 10 do ✓  begin  if NOT(edtCell.Text[I] in ['0' .. '9']) then ✓  begin  edtCell.Clear;  edtCell.SetFocus;  ShowMessage('Cell number must 10 digits only');  exit;  end;  end;  end;  ShowMessage('Welcome to Cell D ' + edtName.Text); ✓  pnlQ1\_2.Enabled := true; ✓  end; | **8** |  |
| **Question 1.2**  procedure TfrmQ1.rgpTypeClick(Sender: TObject);  var  userId: String;  I, iSum: Integer;  begin  // question 1.2  userId := edtName.Text[1]; ✓  userId := userId + (edtName.Text[Pos(' ', edtName.Text) + 1]); ✓✓  userId := UpperCase(userId); ✓  iSum := 0; ✓  for I := 1 to 10 do✓  begin  iSum := iSum + StrToInt(edtCell.Text[I]); ✓✓  end;  userId := userId + IntToStr(iSum); ✓  userId := userId + rgpType.Items[rgpType.ItemIndex][1]; ✓  userId := userId + IntToStr(cmbModel.ItemIndex); ✓  lblUserID.Caption := 'User ID: ' + userId; ✓  end; | **12** |  |
| **Question 1.3**  procedure TfrmQ1.btnBuyClick(Sender: TObject);  var  rCost: real;  begin  // question 1.3  pnlCost.Caption := 'Total Cost : '; ✓  rCost := 0; ✓  if cbxAirtime.Checked then✓  begin  rCost := rCost + sedAirtime.Value; ✓  end;  if cbxData.Checked then✓  begin  sItem := lstData.Items[lstData.ItemIndex]; ✓  iPos := Pos('R',sItem); ✓  rcost := rcost + StrtoFloat(copy(sItem,iPos+1,Length(sItem)-iPos)); ✓✓    end;  if (cbxAirtime.Checked or cbxData.Checked) then✓  begin  rCost := rCost - (rCost \* 0.1); ✓  end;  if NOT (cbxAirtime.Checked and cbxData.Checked) then✓  pnlCost.Caption := 'No options selected'✓  else  pnlCost.Caption := pnlCost.Caption + FloatToStrF(rCost, ffCurrency, 6, 2); ✓✓  end; | **15** |  |
| **Question 1.4**  procedure TfrmQ1.btnBonusClick(Sender: TObject);  var  iChoice, iBonus, iCnt, iTotal : integer;  begin  // question 1.4  redOut.Lines.Clear;  iBonus := 0;  iTotal := 0; ✓  iCnt := 0;  iChoice := StrToInt(InputBox('Choice','Choose a bonus option: 1-Airtime, 2-Data','1')); ✓  if iChoice=1 then✓  redOut.Lines.Add('Airtime Bonus'+#9+'Amount'+#9+'Total') ✓  else ✓  redOut.Lines.Add('Data Bonus'+#9+'Amount'+#9+'Total');  while iTotal<500 do✓✓  begin  inc(iCnt); ✓  iBonus := RandomRange(50,151); ✓ //or Random(100)+50  iTotal := iTotal + iBonus; ✓  redOut.Lines.Add('Bonus Day '+IntToStr(iCnt)+#9+IntToStr(iBonus)+#9+IntToStr(iTotal)); ✓✓✓  end;  //accept other methods of incrementing by 20 days but no hardcoding!  redOut.Lines.Add('Expiry date: '+DateToStr(IncDay(Date,20))) ✓✓  end; | **15** |  |

**Question 2 : possible solution**

|  |  |  |
| --- | --- | --- |
| unit clsCustomer; | **24** |  |
| **// 2.1.1**  interface  uses SysUtils, DateUtils;  type TCustomer = class(TObject)  private  fCallDate : String;  fLineNo := sLineNo;  fStartTime : String; **✓**  fEndTime : String;  fNumber : String;  public  Constructor Create(sCD, sLineNo, sST, sEndT, sNo : String);  procedure setCallDate(cD:String); **✓**  procedure setNumber(no:String);  function callDuration :Integer;  function calcCallCharges : Real;  function calcInternetCharges : Real;  function toString: String;  end;  implementation  { TCustomer }  constructor TCustomer.Create(sCD, sLineNo, sST, sEndT, sNo : String);**✓**  begin  fCallDate := sCD;  fLineNo := sLineNo;  fStartTime := sST; **✓**  fEndTime := sEndT;  fNumber := sNo;  end; | **4** |  |
| **// 2.1.2**  procedure TCustomer.setCallDate(sCD: String);  var  sDay, sMth, sYr : String;  iMth : Integer;  begin  sYr := Copy(sCD,1,4);  sMth := Copy(sCD,6,2); **✓✓**  sDay := Copy(sCD,9,2);  iMth := StrToInt(sMth);  case iMth of  1 : sMth := 'January';  2 : sMth := 'February';  3 : sMth := 'March';  4 : sMth := 'April';  5 : sMth := 'May'; **✓✓**  6 : sMth := 'June';  7 : sMth := 'July';  8 : sMth := 'August';  9 : sMth := 'September';  10: sMth := 'October';  11: sMth := 'November';  12: sMth := 'December';  end; // case  fCallDate := sDay + ' ' + sMth + ' ' + sYr;end; **✓** | **5** |  |
| **// 2.1.3**  procedure TCustomer.setNumber(sNo: String);  begin  fNumber := sNo; **✓**  end; | **1** |  |
| **// 2.1.4**  function TCustomer.callDuration :Integer;  var  times1, times2: TTime;  begin  times1 := StrToTime(startTime); **✓**  times2 := StrToTime(endTime) ; **✓**  result**✓** := SecondsBetween(times1,times2); **✓**// duration of call in seconds  end; | **4** |  |
| **// 2.1.5**  function TCustomer.calcCallCharges : Real; **✓**  begin  result**✓** := (callDuration \* 1.94**✓**) / 60**✓**; // calls charged per minute  end; | **4** |  |
| **// 2.1.6**  function TCustomer.toString: String;  var  tmp:String;  begin  tmp := 'Date' + #9#9#9**✓** + 'Line' + #9 + 'StartTime'+ #9 + 'EndTime' + #9 + 'Duration' + #9 + 'No.Called' + #9 + 'Charge' + #13;  tmp := tmp + **(**fCallDate + #9 + fLineNo + #9 + fStartTime + #9 + fEndTime + #9**) ✓** +  IntToStr(callDuration) **✓** +'sec' + #9#9 + fNumber + #9 +  FloatToStrF(calcCallCharges**✓**, ffCurrency**✓**, 4, 2);  result**✓** := tmp;  end;  end. | **6** |  |
| **unit Question2U;** | **31** |  |
| **//2.2.1**  procedure TfrmQuestion2.btn2\_2\_1Click(Sender: TObject);  var  sNo : String; // number (last value) from text file  begin  AssignFile(q2bill, 'Q2Bill.txt'); **✓**  if fileExists('Q2Bill.txt') **✓** then  Reset(q2bill) **✓**  else  begin  ShowMessage('File not found'); **✓**  exit  end;  try  begin  AssignFile(q2cellbill, 'Q2CellBill.txt'); **✓**  Rewrite(q2cellbill);  end;  except  begin  ShowMessage('Error creating file');  exit  end;  end;  Readln(q2bill, headerLine); **✓** // read header line from text file  // Extract lines containing 10-digit cell nos. from original file Q2Bill.txt and send to new text file Q2CellBill.txt  while not EOF(q2bill) do  begin  Readln(q2bill, sLine); **✓** // extract whole line from text file  iPosSemi:= Pos(';', sLine);  sNo := copy(sLine, iPosSemi+ 1, 10); **✓** // extract no from text file  if Length(sNo) = 10 then**✓**  Writeln(q2cellbill,sLine); **✓** // write sLine to new text file  end; // while  CloseFile(q2cellbill); **✓**  CloseFile(q2bill);  ShowMessage('Q2CellBill.txt created successfully'); **✓**  end; | **12** |  |
| **//2.2.2**  procedure TfrmQuestion2.btn2\_2\_2Click(Sender: TObject);  var  sDateOfCall,  sCD, // callDate from text file  sLineNoTxt,// line number in text file  sST, // startTime of call from text file  sEndT, // endTime of call from text file  sNo, // number (last value) from text file  sCbxLine, // a whole line from combo box  sLineNo // the line no. from combo box  : String;  iPosHash,// position of '#'  iPosDot // position of '.'  : integer;  bLineFound : boolean;  begin  sCbxLine := cbxSelectDate.Items[cbxSelectDate.ItemIndex];**✓**  iPosDot := pos('.',sCbxLine);  sDateOfCall := Copy(sCbxLine, 1, iPosDot-1); **✓** // extract date part  sLineNo := Copy(sCbxLine, iPosDot+1 ,2); **✓** // extract line no.  bLineFound := false;  AssignFile(q2cellbill, 'Q2CellBill.txt'); **✓**  Reset(q2cellbill);  while not EOF(q2cellbill) and not(dateFound)do**✓**  begin  Readln(q2cellbill, sLine); **✓**  posHash := Pos('#', sLine);  sCD := copy(sLine, 1, posHash - 1); // extract date from text file  Delete(sLine, 1, posHash); **✓**  iPosHash := Pos('#', sLine);  sLineNoTxt := copy(sLine, 1, iPosHash - 1);// extract line no. from text file  Delete(sLine, 1, iPosHash); **✓**  iPosHash:= Pos('#', sLine);  sST := copy(sLine, 1, iPosHash - 1); // extract startTime from text file  Delete(sLine, 1, iPosHash); **✓**  iPosSemi:= Pos(';', sLine);  sEndT := copy(sLine, 1, iPosSemi- 1);// extract endTime from text file  Delete(sLine, 1, iPosSemi); **✓**    sNo := sLine; **✓** // extract number from last position of text file  if (sCD = sDateOfCall) and (sLineNo = sLineNoTxt) then **✓✓**  begin  bLineFound:= true; **✓**  end;  end; //while  redCellNoDisplay.Lines.Clear;  if bLineFound then **✓**  begin  objCust := TCustomer.CREATE(sCD, sLineNoTxt, sST, sEndT, sNo);**✓**  objCust.setCallDate(sCD); **✓**  redCellNoDisplay.Lines.Add(objCust.toString); **✓**  end  else  redCellNoDisplay.Lines.Add('Date selected, not in Text File'); **✓**  end; | **19** |  |

|  |  |  |
| --- | --- | --- |
| **unit Question3\_U;** | **45** |  |
| 3.1.  procedure TForm2.btnClearClick(Sender: TObject);  var  r: integer;  c: integer;  begin  for r := 1 to 10 do **✓**  for c := 1 to 10 do **✓**  arrMap[r, c] := '-'; **✓**  arrMap[2, 2] := 'X';  arrMap[2, 9] := 'X'; **✓✓**  arrMap[9, 2] := 'X';  arrMap[9, 9] := 'X';  DisplayMap;  end; | **5** |  |
| 3.2.  function TForm2.blindSpot(x, y: integer): boolean;  var  r, c: integer;  flag: boolean;  begin  flag := true; **✓**  //alternatively learners can check all cells surrounding the point without use //of loop structure  for r := y - 1**✓** to y + 1**✓** do  for c := x - 1 to x + 1 do**✓**  begin  if (arrMap[r, c] **✓**<> '-') then**✓**  flag := false; **✓**  end;  result := flag; **✓**  end; | **8** |  |
| 3.3.  procedure TForm2.btnCoverageClick(Sender: TObject);  var  r, c: integer;  begin  r := spnX.Value;  **✓**  c := spnY.Value;  if (not blindSpot(r, c)) then**✓**  ShowMessage('Your area has LTE coverage.') **✓**  else  ShowMessage('Your area does not have LTE coverage.') **✓**  end; | **4** |  |
| 3.4.  procedure TForm2.btnInstallTowersClick(Sender: TObject);  var  r, c, I: integer;  flag: boolean;  begin  btnClear.Click;  I := 0;  while I < 9 do**✓**  begin  repeat  r := random(8) + 2; **✓range and random method**  c := random(8) + 2;  flag := true; **✓**  if (arrMap[r, c] <> 'X') **✓** and blindSpot(r, c) **✓** then  flag := true  else  flag := false; **✓**  until ((flag = true)); **✓**  arrMap[r, c] := 'X';  inc(I); **✓ //check also if initialization is done i:=0**  end;  DisplayMap;  end; | **8** |  |
| 3.5.  procedure TForm2.btnSubscribeUsersClick(Sender: TObject);  var  r: integer;  c: integer;  begin  for r := 2 to 9 do **✓**  for c := 2 to 9 do **✓**  if arrMap[r, c] = 'X' then**✓**  arrMap[r, c] **✓** := inttostr(random(146) **✓** + 5**✓**);  DisplayMap; **✓**  end; | **7** |  |
| 3.6.  procedure TForm2.btnLeastSubsClick(Sender: TObject);  var  r, c, lowest: integer;  begin  lowest := strtoint(arrMap[2, 2]); **✓**  for r := 2 to 9 do  for c := 2 to 9 do  if arrMap[r, c] <> '-' then **✓**  if (strtoint(arrMap[r, c]) < lowest) then **✓**  begin  lowest := strtoint(arrMap[r, c]);  minPoint := '(' + inttostr(c) + ',' + inttostr(r) + ')'; **✓**  end;  ShowMessage(  'The lowest number of subscribers are registered at coordinates : ' +  minPoint); **✓**  end; | **5** |  |
| 3.7.  procedure TForm2.btnRelocateTowerClick(Sender: TObject);  var  r, c, temp, count: integer;  possible: boolean;  newPos: String;  arrBlindSpots: array [0 .. 100] of String;  begin  count := 0;  possible := false;  for r := 2 to 9 do  for c := 2 to 9 do  if blindSpot(r, c) then **✓**  begin  arrBlindSpots[count] := '(' + inttostr(c) + ',' + inttostr(r) + ')'; **✓**  inc(count);  possible := true;  end;  if possible then**✓**  begin  newPos := arrBlindSpots[random(count)]; **✓**  arrMap[strtoint(newPos[4]), strtoint(newPos[2])] := 'X'; **✓**  arrMap[strtoint(minPoint[4]), strtoint(minPoint[2])] := '-'; **✓**  end  else  ShowMessage(  'No need to relocate tower.You have all points in map covered with LTE.'); **✓**  DisplayMap; **✓**  end; | **8** |  |
| **Grand total** | **150** |  |