education

Department:
Education
PROVINCE OF KWAZULU-NATAL

JUNE EXAMINATION 2017
INFORMATION TECHNOLOGY
GRADE 12
PAPER 1

MARKING GUIDE AND POSSIBLE SOLUTIONS

QUESTION 1

NO	TASKS	Max	Mark
1.1	Extracts name from edtName for processing ✓ Extracts Surname from edtSurname for processing ✓ Conditional Statement to evaluate cmbType ✓ Sets “#” symbol for Index 0 ✓ Sets “@” symbol for Index 1 ✓ Extracts Age from sedAge ✓ Inverts age digits ✓✓ Generates code by combining: First two letters of name✓ Last two letters of surname✓✓✓ Type symbol✓ Age digits inverted✓ Displays generated code in edtQ1_1. ✓	15	
1.2	Extracts value from sedNoOfPlayers for processing✓ Conditional structure to test rgbEntryType for Index 0✓ Assigns Price to 1000 ✓ Conditional structure to test rgbEntryType for Index 1✓ Assigns Price to 850✓ Tests Number of Players for divisibility by 2✓ Calculates total cost correctly✓ Displays total cost to edtQ1_2✓ formatted✓. Error message if Num of Players not divisible by 2. ✓	10	
1.3	Checks if players.txt exists ✓ Displays error message and stops execution if not✓ Assigns file to players.txt✓ Opens file for reading✓ Loops to end of text file✓ Extracts line from text file✓ Splits line from text file based on delimiter✓✓ Tests correct extract from line with Input GamerTag (from edtGamerTag) ✓ If found: Displays Registration Num with caption✓ Player Tag with caption ✓	15	

	Gender with caption✓ Evaluates Registration Status and displays "Registered" OR "Payment Outstanding". ✓ Closes file✓ Displays "User Not Found" if entry is not found. ✓		
1.4	Extracts input data from edtReqTag and edtSAID for processing ✓ Sets a Boolean Flag for testing validity of input / OR uses an alternative structure correctly✓ Tests Length with correct range✓✓ Loops from 1 to Length of input tag for evaluation ✓ Tests each letter for validity changing the flag variable, if necessary✓ Error message✓ if validation failed ✓ Counter variable to count number of lines in text file assigned to 0. ✓ Text file assigned and opened✓ Loop through text file✓ Counter increased✓ Counter increased by 1✓ Extracts characters 7-10 from ID✓ Evaluates extracted data ✓ Sets F correctly✓ Sets M correctly✓ Combines data to form output string✓ Writes output string to text file✓ Displays output string in edtQ1_4✓	20	
		[60]	

POSSIBLE SOLUTION : QUESTION 1

```
procedure TForm1.btnQ1_1Click(Sender: TObject);
var
   sAge, sCode, sName, sSurname, sType, sInv : String;

begin
  sName := edtName.Text; ✓
  sSurname := edtSurname.Text; ✓
  if cmbType.ItemIndex = 0 then✓
  begin
     sType := '#'; ✓
  end
  else
  begin
     sType := '@'; ✓
  end;

  sAge := IntToStr(sedAge.Value); ✓

  sAge := sAge[2] ✓ + sAge[1]; ✓

  sCode := sName[1] + sName[2] ✓ + copy(sSurname✓,
 (Length(sSurname)-1✓), 2✓) + sType✓ + sAge✓;

  edtQ1_1.Text := sCode; ✓
end;

procedure TForm1.btnQ1_2Click(Sender: TObject);
var
   iNum, iPrice : Integer;
   rTotal : Real;
begin
   iNum := sedNoOfPlayers.Value; ✓

   if rgbEntryType.ItemIndex = 0 then✓
   begin
     iPrice := 1000; ✓
   end
   else✓
   begin
     iPrice := 850; ✓
   end;

   if iNum mod 2 = 0 then✓
   begin
     rTotal := iPrice * iNum; ✓
     edtQ1_2.Text := FloatToStrF(rTotal✓, ffCurrency, 8, 2) ✓;
   end
```

```
    else
    begin
      showMessage('Number of players must be even'); ✓
    end;

end;

procedure TForm1.btnQ1_3Click(Sender: TObject);
var
  t : TextFile;
  sSplit : TStrings;
  s : String;
begin

  if not fileexists('players.txt') then ✓
  begin
    showMessage('File Not Found'); ✓
  end
  else
  begin
    AssignFile(t, 'players.txt'); ✓
    Reset(t); ✓

    redQ1_3.Clear;

    while not eof(t) do✓
    begin
      ReadLn(t, s); ✓
      sSplit := TStringList.Create; ✓

      ExtractStrings(['#'],[],PChar(s),sSplit); ✓

      if edtGamerTag.Text = sSplit[1] then✓
      begin
        redQ1_3.SelText := ('Registration Number: '+sSplit[0] +
                         #13+✓
                          'Player Tag: '+ sSplit[1] + #13+✓
                          'Gender: '+sSplit[2] +  #13+✓
                          'Registration Status: ');   ┐
        if sSplit[3] = 'PAID' then                    │
           redQ1_3.SelText := 'Registered'            ├      ✓
        else                                          │
           redQ1_3.SelText := 'Payment Outstanding';  ┘

      end;

    end;
    CloseFile(t); ✓
    if redQ1_3.Text = '' then                 }
       redQ1_3.Text := 'User Not Found';}  ✓
  end;
end;

procedure TForm1.btnQ1_4Click(Sender: TObject);
const
   alphabet = ['a'..'z','A'..'Z'];
var
   sReqTag, sGenTag, sID, sGender, s : String;
   iGenCode, i, c : Integer;
   bCheck : Boolean;
   t : TextFile;
begin
   sReqTag := edtReqTag.Text; }      ✓
   sID := edtSAID.Text;       }

   bCheck := TRUE; ✓

   if (Length(sReqTag) >= 6) ✓ AND (Length(sReqTag) <= 14) ✓  then
   begin
      for i := 1 to Length(sReqTag) do✓
      begin
        if not(sReqTag[i] in alphabet) then bCheck := FALSE; ✓
      end;
   end
   else
   begin
     bCheck := FALSE;
   end;

   if not bCheck then ✓
   begin
     showMessage('Gamer Tag does not meet requirements'); ✓
   end
   else
   begin
      c := 0; ✓
      AssignFile(t, 'players.txt');
      Reset(t); ✓
      while not eof(t) do✓
      begin
        ReadLn(t, s);
        inc(c); ✓
      end;
      CloseFile(t);

      inc(c); ✓

      iGenCode := StrToInt(copy(sID, 7, 4)); ✓
      if iGenCode < 5000 then✓
         sGender := 'F'✓
      else
         sGender := 'M'; ✓

      sGenTag := IntToStr(c)+'#'+sReqTag+'#'+sGender+'#NOTPAID'; ✓
      Append(t);
      WriteLn(t, sGenTag); ✓
      CloseFile(t);
      edtQ1_4.Text := sGenTag; ✓
   end;
end;
```

ALTERNATIVE SOLUTION : QUESTION 1

```
procedure TForm1.btnQ1_1Click(Sender: TObject);
var
  sName, sSurname, sType, sFirst2, sLast2: String;
  iAge, iLen, iRev: Integer;
  cUser: char;
begin
  { Question 1.1 }
  sName := edtName.Text✓;
  sSurname := edtSurname.Text✓;
  sType := cmbType.items[cmbType.ItemIndex];
  iAge := sedAge.Value✓;
  sFirst2 := copy(sName, 1, 2) ✓;
  iLen := length(sSurname);
  sLast2 := copy(sSurname, iLen - 1) ✓;
  if sType = 'Internal' then✓
  begin
    cUser := '#'✓
  end
  else
  begin
    cUser := '@'; ✓
  end;
  iRev := (iAge mod 10) * 10✓ + iAge div 10✓;
  edtQ1_1.Text := sFirst2 + sLast2 + cUser✓ + IntToStr(iRev) ✓;

end;

procedure TForm1.btnQ1_2Click(Sender: TObject);
var
  iIndex, iNumPlayers: Integer;
  rAmount, rTotal: Real;
begin
  { Question 1.2 }
  iIndex := rgbEntryType.ItemIndex;
  rAmount := 0;
  rTotal := 0;
  case iIndex of✓
    0:
      rAmount := 1000; ✓
    1:
      rAmount := 850.00✓;
  end;
  iNumPlayers := sedNoOfPlayers.Value✓;
  if iNumPlayers mod 2 = 0 then✓
  begin
    rTotal := iNumPlayers * rAmount✓;
    edtQ1_2.Text := FloatToStrF(rTotal✓, ffCurrency, 8, 2) ✓;
  end
  else
  begin
    ShowMessage('Number of players must be even') ✓;
  end;
end;

procedure TForm1.btnQ1_3Click(Sender: TObject);
var
  tName: Textfile;
  sLine, sTag, sGamerTag, sPaid, Snum, sGender, sStatus: String;
  iNum, iPos: Integer;
  cGender: char;
  bFlag: Boolean;
begin
  { Question 1.3 }
  try
    begin
      sGamerTag := UpperCase(edtGamerTag.Text);
      AssignFile(tName, 'Players.txt'); ✓
      Reset(tName); ✓
      bFlag := false;
      while (NOT EOF(tName)) ✓ do
      begin
        Readln(tName, sLine) ✓;
        // redQ1_3.Lines.Add(sLine);
        iPos := Pos('#', sLine); ✓
        Snum := copy(sLine, 1, iPos - 1); ✓
        Delete(sLine, 1, iPos); ✓
        iPos := Pos('#', sLine);
        sTag := copy(sLine, 1, iPos - 1); ✓
        // redQ1_3.Lines.Add(sTag);
        Delete(sLine, 1, iPos); ✓
        iPos := Pos('#', sLine); ✓
        sGender := copy(sLine, 1, 1); ✓
        sPaid := copy(sLine, iPos + 1); ✓
        sStatus := 'Registered';
        if sPaid = 'NOTPAID' then
        begin
          sStatus := 'Payment Outstanding'; ✓
        end;
        if sGamerTag = sTag then
        begin
          bFlag := true;
          redQ1_3.Lines.Clear;
          redQ1_3.Lines.Add('Registration Number ' + Snum);
          redQ1_3.Lines.Add('Player Tag ' + sTag);
          redQ1_3.Lines.Add('Gender  ' + sGender);
          redQ1_3.Lines.Add('Registration Status ' + sStatus);
        end
      end;
      close(tName); ✓
    end;
  except✓
    begin
      ShowMessage('File not found') ✓;
    end;
  end;
  if bFlag = false then
  begin
    redQ1_3.Lines.Clear;
    redQ1_3.Lines.Add('User Not Found');
  end;

end;

procedure TForm1.btnQ1_4Click(Sender: TObject);
var
  sReqTag, sGenTag, sID, sGender, s: String;
  iGenCode, i, c: Integer;
  bCheck: Boolean;
  t: Textfile;

begin
  { Question 1.4 }
  sReqTag := edtReqTag.Text;
  sID := edtSAID.Text; ✓
  bCheck := true;
  if (length(sReqTag) >= 6) ✓ AND (length(sReqTag) <= 14) ✓ then
  begin
    for i := 1 to length(sReqTag) ✓ do
    begin
      if not(sReqTag[i] in ['a' .. 'z', 'A' .. 'Z') then
        bCheck := false; ✓
    end;
  end
  else
  begin
    bCheck := false;
  end;

  if not bCheck then✓
  begin
    ShowMessage('Gamer Tag does not meet requirements'); ✓
  end
  else
  begin
    c := 0; ✓
    AssignFile(t, 'players.txt');
    Reset(t); ✓
    while not EOF(t) do✓
    begin
      Readln(t, s);
      inc(c); ✓
    end;
    CloseFile(t);
    inc(c); ✓
    iGenCode := StrToInt(copy(sID, 7, 4)) ✓;
    if iGenCode < 5000 then
      sGender := 'F'✓
    else
      sGender := 'M'; ✓

 sGenTag := IntToStr(c) + '#' + sReqTag + '#' + sGender +
'#NOTPAID'; ✓
    Append(t);
    WriteLn(t, sGenTag); ✓
    CloseFile(t);
    edtQ1_4.Text := sGenTag; ✓
  end;

end;

End.
```

QUESTION 2

NO	TASKS	Max	Mark
2.1.1	Declares all 4 attributes correctly✓✓✓✓ With the most suitable data types used for all attributes✓	5	
2.1.2	Declares Constructor in Interface✓ Implements Constructor correctly (header) ✓ by assigning received parameters to attributes. ✓✓✓✓	6	
2.1.3	Declares mutator under Interface✓ Implements mutator correctly (header) ✓ Assigns received parameter to attribute ✓	3	
2.1.4	Declares accessor under Interface✓ Implements accessor correctly (header) ✓ Sends Score attribute correctly✓	3	
2.1.5	Declares CalcAve function correctly under Interface✓ Implements CalcAve correctly (header) ✓ Result calculated using correct formula✓	3	
2.1.6	Declares ProcessFoul under Interface✓ Implements ProcessFoul correctly (header) ✓ Conditional statement to evaluate FoulStatus attribute✓ Score decremented✓ by correctly calculated value✓	5	
2.1.7	Declares toString under Interface✓ Implements toString correctly (header) ✓ Outputs Team Name with correct caption + #13✓ Players with correct caption and new line✓ Score with converted value, caption and new line✓ Evaluates Foul Status ✓ Outputting "Yes" with caption if TRUE✓ "No" with caption if FALSE✓ Returns output correctly. ✓	9	
		(34)	
2.2.1	Extracts data from all input components correctly✓✓✓✓ Calls Constructor correctly✓ with arguments in the correct order. ✓ Confirmation Message✓	7	
2.2.2	Mutator called correctly ✓ Sending value from CheckBox chbUpdate✓	2	
2.2.3	ProcessFoul method called correctly✓	1	
2.2.4	Calls Accessor correctly ✓displaying value✓	2	
2.2.5	Calls Average function✓ displaying value correctly✓	2	
2.2.6	Calls toString function✓, displaying value in redOutput✓	2	
		(16)	
		[50]	

POSSIBLE SOLUTION: QUESTION 2

Question 2.1

```
unit clsTeam;

interface

type
  TTeam = class(TObject)

private
{2.1.1}
  FTeamName : String; ✓
  FNoPlayers : Integer; ✓              ✓
  FScore : Integer; ✓
  FFoulStatus : Boolean; ✓

public

{2.1.2}
constructor CREATE(sTeamName : String; iNoPlayers, iScore :
Integer; bFoulStatus : Boolean); ✓

{2.1.3}
procedure setFoulStatus(bFoulStatus : Boolean); ✓

{2.1.4}
function getScore : Integer; ✓

{2.1.5}
function calcAve : Real; ✓

{2.1.6}
procedure processFoul; ✓

{2.1.7}
function toString : String; ✓

end;

implementation
uses SysUtils;

{2.1.2}
constructor TTeam.CREATE(sTeamName : String; iNoPlayers, iScore :
Integer; bFoulStatus : Boolean); ✓
begin
  FTeamName := sTeamName; ✓
  FNoPlayers := iNoPlayers; ✓
  FScore := iScore; ✓
  FFoulStatus := bFoulStatus; ✓
end;

{2.1.3}
procedure TTeam.setFoulStatus(bFoulStatus : Boolean); ✓
begin
  FFoulStatus := bFoulStatus; ✓
end;

{2.1.4}
function TTeam.getScore : Integer; ✓
begin
  Result := FScore; ✓
end;

{2.1.5}
function TTeam.calcAve : Real; ✓
begin
  Result := FScore / FNoPlayers; ✓
end;

{2.1.6}
procedure TTeam.processFoul; ✓
begin
  if FFoulStatus = TRUE then✓
  begin
    FScore := FScore - ✓Round(FScore * 0.1) ✓;
  end;
end;

{2.1.7}
function TTeam.toString : String; ✓
var
   sOutput : String;
begin
   sOutput := 'TEAM:'+#9+FTeamName+#13+✓
              'PLAYERS:'+#9+IntTOStr(FNoPlayers)+#13+✓
              'SCORE:'+#9+IntToStr(FScore)+#13+✓
              'FOULS?'+#9;
   if FFoulStatus = TRUE then✓
      sOutput := sOutput + 'YES'✓
   else
      sOutput := sOutput + 'NO'; ✓

   Result := sOutput; ✓
end;

end.
```

Question 2.2

```
procedure TForm1.btnCreateClick(Sender: TObject);
var
  sTeam : String;
  iScore, iPlayers : Integer;
  bFouls : Boolean;
begin
  sTeam := edtTeamName.Text; ✓
  iScore := sedScore.Value; ✓
  iPlayers := sedPlayers.Value; ✓
  bFouls := chbFouls.Checked; ✓

  objTeam := TTeam.CREATE✓ (sTeam, iPlayers, iScore, bFouls) ✓;

  showMessage('Object Created'); ✓
end;

procedure TForm1.btnUpdateClick(Sender: TObject);
begin
  objTeam.setFoulStatus✓ (chbUpdate.Checked✓);
end;

procedure TForm1.btnProcessClick(Sender: TObject);
begin
  objTeam.processFoul; ✓
end;

procedure TForm1.btnScoreClick(Sender: TObject);
begin
  showMessage('Current Score: '+✓IntToStr(objTeam.getScore) ✓);
end;

procedure TForm1.btnAverageClick(Sender: TObject);
begin
  showMessage('Average Score: '+✓FloatToStr(objTeam.calcAve) ✓);
end;

procedure TForm1.btnDisplayClick(Sender: TObject);
begin
  redOutput.Text := ✓ objTeam.toString✓;
end;
```

QUESTION 3

NO	TASKS	Max	Mark
3.1	Loop✓ to Length of arrNames✓ Generates random number for row in correct range✓ Generate random number for col in correct range✓ Structure to ensure row / col combination has not been chosen already (NOTE: This can be achieved in a variety of different ways – Trace learner solutions to check viability of solution and allocate marks at your discretion) ✓✓✓✓✓✓✓ Assign arrSeating ✓to value from arrNames✓ (allocate marks if array and index is used correctly) DispData method called ✓	14	
3.2	Extracts data from sedRow✓ Extracts data from sedColumn✓ Checks ✓if seat is vacant✓ If not vacant, displays “Booked” message✓ If seat is vacant✓, gets learner’s name from Dialogue Box✓ Assigns arrSeating [Correct Indices] ✓ to received name ✓ DispData method called✓	10	
3.3	Extracts Price from sedPrice✓ Sets TotalEarnings variable to 0✓ Loop from 1 to 5 (Loop A) ✓ Sets arrEarnings[correct index] to 0✓ Loops from 1 to 6 (Loop B) ✓ Checks✓ if seat at pos [A][B] is taken✓ If taken, adds Price value to ✓ arrEarnings [correct index] ✓ Increases ✓TotalEarnings value correctly✓ *(NOTE: Some learners may do this separately outside of the nested loops; with its own loop)* Decreases price correctly (-10%)✓✓ DispData method called✓ Displays TotalEarnings formatted as currency✓ with suitable caption✓	16	
		[40]	
GRAND TOTAL:		**150**	

POSSIBLE SOLUTION: QUESTION 3

```
procedure TForm1.btnSeatRandomClick(Sender: TObject);
var
  i, j, c : Integer;
begin

  for c :=✓ 1 to 10 do✓
  begin

    i := Random(5) + 1; ✓
    j := Random(6) + 1; ✓

    while not✓ (arrSeating[i][j] ✓ = '#') do✓
    begin
        i := ✓Random(5) + 1; ✓
        j := ✓Random(6) + 1; ✓
    end;

    arrSeating[i][j] ✓:= arrNames[c]; ✓

  end;

  dispData; ✓
end;
```

Repeat loop may be used in many learner solutions instead of the While..Do

```
procedure TForm1.btnBookClick(Sender: TObject);
var
  i, j : Integer;
  sName : String;
begin
  i := sedRow.Value; ✓
  j := sedColumn.Value; ✓

  if arrSeating[i][j] ✓<> '#' then✓
  begin
    showMessage('Seat already taken'); ✓
  end
  else✓
  begin
    sName :=✓ InputBox('Q3','Enter name',''); ✓
    arrSeating[i][j] := sName; ✓
    dispData; ✓
  end;

end;
```

```
procedure TForm1.btnCalcIncomeClick(Sender: TObject);
var
  i, j : Integer;
  rPrice, rTotal : Real;
begin
  rPrice := sedPrice.Value; ✓
  rTotal := 0; ✓

  for i := 1 to 5 do✓
  begin

    arrEarnings[i] := 0; ✓

    for j := 1 to 6 do✓
    begin

      if arrSeating[i][j] ✓ <> '#' then✓
         arrEarnings[i] :=✓ arrEarnings[i] + rPrice; ✓

    end;
    rTotal := ✓rTotal + arrEarnings[i]; ✓
    rPrice := ✓rPrice - (rPrice * 0.1); ✓

  end;

  dispData; ✓
  redOutput.Lines.Add(#13+'Total earnings: ✓ '+FloatToStrF(rTotal,
ffCurrency, 8, 2)); ✓
end;
```