basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

NATIONAL SENIOR CERTIFICATE

GRADE/*GRAAD* 12

INFORMATION TECHNOLOGY P1/*INLIGTINGSTEGNOLOGIE V1*

FEBRUARY/MARCH/*FEBRUARIE/MAART* (1) 2012

MEMORANDUM

MARKS/*PUNTE*: 120

This memorandum consists of 14 pages./
Hierdie memorandum bestaan uit 14 bladsye.

Copyright reserved/*Kopiereg voorbehou*

GENERAL INFORMATION:

- **Pages 2–11 contain the memoranda of possible solutions for QUESTIONS 1 to 3 in programming code.**
- **Pages 12–14 contain ADDENDA A to C which include a marking grid for each question.**
- **Copies of the appropriate ADDENDA should be made for each learner to be completed during the marking session.**

QUESTION 1: PROGRAMMING AND DATABASE

```
unit Question1_U;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, DB, ADODB, Grids, DBGrids, ExtCtrls, Buttons;

type
  TfrmRec = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    btnA: TButton;
    btnB: TButton;
    btnC: TButton;
    btnD: TButton;
    btnE: TButton;
    btnF: TButton;
    btnG: TButton;
    BitBtn1: TBitBtn;
    qryQOne: TADOQuery;
    tblRecAg: TDataSource;
    grdRec: TDBGrid;

    procedure btnAClick(Sender: TObject);
    procedure btnBClick(Sender: TObject);
    procedure btnCClick(Sender: TObject);
    procedure btnDClick(Sender: TObject);
    procedure btnEClick(Sender: TObject);
    procedure btnFClick(Sender: TObject);
    procedure btnGClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmRec: TfrmRec;

implementation

{$R *.dfm}
```

```
procedure TfrmRec.btnAClick(Sender: TObject);
begin
  qryQOne.Active := False;                                    //QUESTION 1.1

  qryQOne.SQL.Text :=  'SELECT SchoolName, Category, Balance, LastPayment ✓' +
                       'FROM tblSchools✓ ORDER BY Category✓, LastPayment ✓';
  qryQOne.Active := True;
end;                                                                    (4)
//=========================================================================

procedure TfrmRec.btnBClick(Sender: TObject);                 //QUESTION 1.2
begin
  qryQOne.Active := False;
  qryQOne.SQL.Text :=  'SELECT SchoolName, Balance, LastPayment FROM
                        tblSchools ✓ WHERE (Balance > 0) ✓
                        AND ✓ (#2011/11/25#✓-lastpayment ✓> 60✓)';

{ALTERNATELY: WHERE
(Balance>0)AND(DateDiff("d",[LastPayment],"2011/11/25")>60)';
    qryQOne.Active := True;
end;                                                                    (6)
//=========================================================================
                                                              //QUESTION 1.3
procedure TfrmRec.btnCClick(Sender: TObject);
begin
  qryQOne.Active := False;
  qryQOne.SQL.Text := 'SELECT Officer, Count✓(LogTime) AS [CallsPeakHours] ' +
                        'FROM tblCallouts '✓ +
                        'WHERE (HOUR✓(LogTime) Between✓ 15 and 24✓)  ' +
                        'GROUP BY Officer'✓;
  qryQOne.Active := True;
end;                                                                    (6)
//=========================================================================

procedure TfrmRec.btnDClick(Sender: TObject);                 //QUESTION 1.4
begin
  qryQOne.Active := False;
  qryQOne.SQL.Text := 'SELECT SchoolName, Reason , ' +
                      'Count(LogDate) AS [Number of visits]✓' +
                      'FROM tblSchools, tblCallouts ✓ ' +
                   'WHERE (tblSchools.SchoolNumber = tblCallouts.SchoolNumber)✓
      ' +
                      ' AND (Year(LogDate) = 2011)✓   ' +
                      'Group by SchoolName✓, Reason✓ ';
  qryQOne.Active := True;
end;                                                                    (6)
//=========================================================================

procedure TfrmRec.btnEClick(Sender: TObject);                 //QUESTION 1.5
begin
  qryQOne.Active := False;
  qryQOne.SQL.Text := 'SELECT Format(SUM✓(Balance),"Currency") AS [Total
     amount] ✓, Format(AVG✓(Balance),"Currency") AS [Average amount] ' +
                      'FROM tblSchools ' +
                      'WHERE (Category = "S")✓';
  qryQOne.Active := True;
end;                                                                    (4)
//=========================================================================
```

BOTH FORMAT functions: ONLY ONE mark!

```
                                                                    //QUESTION 1.6
procedure TfrmRec.btnFClick(Sender: TObject);
var
  iX : Integer;
begin
  qryQOne.Active := False;
  qryQOne.SQL.Text := 'UPDATE tblSchools  '✓ +
                      'SET✓ Balance = Balance - Instalments✓, ' +
                      'LastPayment = #25/11/2011# ✓' +
                        'WHERE Category = "P" ✓';
  qryQOne.ExecSQL;
  MessageDlg('Records Processed Successfully',mtInformation,[mbok],0);

end;                                                                          (5)

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

procedure TfrmRec.btnGClick(Sender: TObject);                 //QUESTION 1.7
var
  iX : Integer;
begin
  qryQOne.Active := False;
  qryQOne.SQL.Text := 'DELETE✓ FROM tblCallouts✓ WHERE Year✓(LogDate) =
2010✓';
  qryQOne.ExecSQL;
  MessageDlg('Records Processed Successfully',mtInformation,[mbok],0);
end;                                                                         (4)
//============================================================================
end.
```

[35]

QUESTION 2: OBJECT-ORIENTED PROGRAMMING

```
unit uHouseCheckXXXX;

interface

uses Classes, SysUtils;

//=====================================================================
// Q 2.1.1                                    (3)
type THouseCheck = class✓
  private✓
    fServices : String; ✓
    fNumDays : integer;

public
    constructor Create(iNumber : integer; sServ : String);
    function isServiceNeeded(cService : char) : boolean;
    function getServicesForDay(iDay : integer) : String;
    function toString : String;
    function getTotalCost : real;
    procedure extendDays(iDays : integer);
end;

implementation
//=====================================================================
// Q 2.1.2                                    (3)

constructor THouseCheck.Create✓ (iNumber: integer; sServ: String✓);
begin
  fNumDays := iNumber;      }
  fServices := sServ;       }   ✓
end;


//=====================================================================
// Q 2.1.3                                    (5)

function THouseCheck.isServiceNeeded(cService: char✓): boolean✓;
begin
  if (Pos(cService, fServices) > 0) ✓✓ then
    Result := true          }
  else                      }  ✓
    Result := false;        }
end;

//=====================================================================
```

Q 2.1.1
(1) Correct class declaration
(1) Attributes declared private
(1) Correct name & data type for both the local attributes

Q 2.1.2
(1) Constructor declared correctly
(1) Correct parameters
(1) Values assigned correctly to attributes

Q 2.1.3
(1) Method returns Boolean
(1) Receives character as parameter
(2) Searches for character in string
(1) Returns true or false accordingly

***//* Q 2.1.4** **(7)**

```
function THouseCheck.getServicesForDay(iDay: integer✓): String✓;
var
  sServStr : String;
begin
  sStr := '';

  if isServiceNeeded('A') ✓ then
    sServStr := sServStr + 'Feed pets' + #9;
  if (isServiceNeeded('C') AND (iDay MOD 2 = 0)) ✓ then
    sServStr := sServStr + 'Check windows and doors' + #9; ✓
  if (isServiceNeeded('P') AND (iDay MOD 3 = 0)) ✓ then
    sServStr := sServStr + 'Empty post box';

  if sServStr = '' then
    sServStr := 'No services for the day'; ✓

  Result := sServStr;
end;
```

Q 2.1.4
(1) Method receives an integer parameter
(1) Returns a formatted string
(1) Checks if service is required
(2) on a particular day
(1) Concatenates service onto return string
(1) Checks for "No services" condition

```
//============================================================================
```

***//* Q 2.1.5** **(6)**

```
function THouseCheck.toString: String; ✓
var
  i : integer;
  sStr : String;
begin
  sStr := '';

  for i := 1 to fNumDays do✓
    begin
      sStr := sStr✓ + IntToStr(i) ✓ + #9 + getServicesForDay(i) ✓ + #13; ✓
    end;

  Result := sStr;
end;
```

Q 2.1.5
(1) Method called toString which returns a string
(2) For loop for number of days
(1) Joins day number and service for that day
(1) Using getServicesForDay method
(1) With a tab and newline

```
//============================================================================
```

***//* Q 2.1.6** **(5)**

```
function THouseCheck.getTotalCost: real; ✓
var
  rTotalCost : real;
begin
  rTotalCost:= 0; ✓

  if isServiceNeeded('A') then✓
       rTotalCost := rTotalCost + fNumDays * 5;
    if isServiceNeeded('C') then
     rTotalCost:= rTotalCost + fNumDays DIV 2 * 6; ✓
    if isServiceNeeded('P') then
     rTotalCost:= rTotalCost + fNumDays DIV 3 * 4; ✓

  Result := rTotalCost;
end;
```

Q 2.1.6
(1) Method returns a real number
(1) Initialises a total variable to zero
(1) Tests if a service is needed
(2) Adds to cost for a service using correct formula

```
//============================================================================
```

***//* Q 2.1.7** **(3)**

```
procedure THouseCheck.extendDays(iDays: integer); ✓
begin
  Inc(fNumDays ✓, iDays); ✓
end;
```

Q 2.1.7
(1) Method declared correctly
(1) Adds number of days
(1) to days attribute

```
unit Question2XXXX_U;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Menus, StdCtrls, ComCtrls, uHouseCheckXXXX;

type
  TfrmQuest2 = class(TForm)
    MainMenu1: TMainMenu;
    OptionA1: TMenuItem;
    OptionB1: TMenuItem;
    OptionC1: TMenuItem;
    Quit1: TMenuItem;
    redOutput: TRichEdit;
    OptionD1: TMenuItem;
    procedure Quit1Click(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure OptionA1Click(Sender: TObject);
    procedure OptionB1Click(Sender: TObject);
    procedure OptionC1Click(Sender: TObject);
    procedure OptionD1Click(Sender: TObject);private
    { Private declarations }
  public
    { Public declarations }
  end;
//=========================================================================
```

***//* Q 2.2.1** **(4)**

```
var
  frmQuest2: TfrmQuest2;
  house : THouseCheck; ✓

procedure TfrmQuest2.FormActivate(Sender: TObject);
var
  iDays : integer;
  sServices : String;
begin
  iDays := StrToInt(InputBox('Input days', 'How many days will the client
              be away for?', ''));
  sServices := Uppercase(InputBox('Input services', 'Which services are
                required?',  ''));                                        ✓
  house := THouseCheck.Create✓(iDays, sServices✓);
end;

//=========================================================================
```

Q 2.2.1
(1) THouseCheck object declared correctly
(1) User enters a string and integer value
(1) Calls constructor method
(1) with correct parameters

***//* Q 2.2.2** **(2)**

```
procedure TfrmQuest2.OptionA1Click(Sender: TObject);
begin
  redOutput.Lines.Clear;
  redOutput.Lines.Add('Day' + #9 + 'Services'); ✓
  redOutput.Lines.Add(house.toString); ✓
end;
//=========================================================================
```

Q 2.2.2
(1) Displays a suitable heading
(1) Calls the toString method and displays result

***//* Q 2.2.3** **(2)**

```
procedure TfrmQuest2.OptionB1Click(Sender: TObject);
begin
  redOutput.Lines.Clear;
  redOutput.Lines.Add('Total cost: '✓+
     FloatToStrF(house.getTotalCost,ffCurrency,12,2)✓);
  redOutput.Lines.Add('');
end;

//============================================================================
```

Q 2.2.3
(1) Calls getTotalCost method
(1) Displays result as a currency with suitable label

***//* Q 2.2.4** **(4)**

```
procedure TfrmQuest2.OptionC1Click(Sender: TObject);
var
  iDay : integer;
begin


  iDay := StrToInt(InputBox('Input day', 'Enter the day you want to see the
services for', ''));✓
  redOutput.Lines.Clear;
  redOutput.Lines.Add('Services required for day ' + IntToStr(iDay) + ' : ' +
house.getServicesForDay✓(iDay✓));✓
end;

//============================================================================
```

Q 2.2.4
(1) User enters day as an integer
(1) Calls getServicesForDay method
(1) with parameter and
(1) Displays the result

***//* Q 2.2.5** **(2)**

```
procedure TfrmQuest2.OptionD1Click(Sender: TObject);
var
  iMoreDays : integer;
begin
  redOutput.Lines.Clear;
  iMoreDays := StrToInt(InputBox('Input days', 'How many more days will the
person be away?', ''));✓
  house.extendDays(iMoreDays);✓
end;

//============================================================================
```

Q 2.2.5
(1) User enters days as an integer
(1) Calls extendDays method with parameter

[46]

QUESTION 3: PROGRAMMING

```
unit Question3_U;

interface

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

type
  TfrmQuest3 = class(TForm)
    redOutput: TRichEdit;
    pnlOptions: TPanel;
    btnA: TButton;
    btnB: TButton;
    BitBtn1: TBitBtn;
    procedure FormCreate(Sender: TObject);
    procedure btnAClick(Sender: TObject);
    procedure btnBClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmQuest3: TfrmQuest3;
  arrInfo  : array[1..12] of String[30];
implementation



{$R *.dfm}

procedure TfrmQuest3.FormCreate(Sender: TObject);
begin
  arrInfo[1]:= 'HDRDBRHRHDdRHHDDrhbRR';
  arrInfo[2]:= 'RrBHDrBrRHBBDRHDdBRRhb';
  arrInfo[3]:= 'BBRRRbDBRrBBHHRBBB';
  arrInfo[4]:= 'RBBRBRdBRDBDRRRBbRRRRbBDHHH';
  arrInfo[5]:= 'RHDBBBHRBB';
  arrInfo[6]:= 'RbrBrBDdRrBb';
  arrInfo[7]:= 'bRRbhbdB';
  arrInfo[8]:= 'hbB';
  arrInfo[9]:= 'BHRrB';
  arrInfo[10]:= 'HhbrbBHhRrdrbBDRRRBB';
  arrInfo[11]:= 'BBBbrrRRdrHRBbBRdDBRBDRrBBDRdB';
  arrInfo[12]:= 'BRBrHRBBRrhHHrrRBDbBDbRRrRRBBDRBDDRHhHD';

  redOutput.Paragraph.TabCount := 3;
  redOutput.Paragraph.Tab[0] := 1;
  redOutput.Paragraph.Tab[1] := 30;
  redOutput.Paragraph.Tab[2] := 300;

end;

// ============================================================================
```

// QUESTION 3.1

```
procedure TfrmQuest3.btnAClick(Sender: TObject);
var
  K, M, iCount, iTotalRob, iTotalCrime  :integer;
  sNewString, sMonth    :String;
  rPercentage :real;

begin
  redOutput.Lines.Add('Month' + #9 + 'Statistics'
                  + #9 + 'Armed Robberies'); ✓
  iTotalRob := 0;
  iTotalCrime := 0; ✓

  for K := 1 to 12 do✓ // per month
    begin
      iCount := 0; ✓
      sNewString := IntToStr(K)+ #9; ✓
      for M := 1 to length(arrInfo[K]) do✓
        begin
        case  upcase✓ (arrInfo[K][M]) ✓ of

        'B','D','H'✓:sNewString := sNewString ✓+
                              lowercase✓ (arrInfo[K][M]); ✓
        'R':begin✓
              sNewString := sNewString + upcase✓ ( arrInfo[K][M]); ✓
              iCount := iCount + 1; ✓
            end;
        end;

          {Alternative:
          if arrInfo[K][M] in ['B', 'D','H'] then
             sNewString := sNewString + lowercase(arrInfo[K][M])
          else
            if arrInfo[K][M] in ['b', 'd','h'] then
             sNewString := sNewString + arrInfo[K][M]
           else
            if  arrInfo[K][M] in ['R', 'r'] then
            begin
             sNewString := sNewString + upcase( arrInfo[K][M]);
             iCount := iCount + 1;
            end; }

        end;
        iTotalRob := iTotalRob + iCount; ✓
        iTotalCrime := iTotalCrime + length(arrInfo[K]); ✓
        sNewString :=  #9 + sNewString + #9 + intToStr(iCount); ✓
        redOutput.Lines.Add(sNewString); ✓
     end;
     rPercentage := iTotalRob / iTotalCrime * 100; ✓
     redOutput.Lines.Add(' ');
     redOutput.Lines.Add('The percentage armed robberies: ' ✓+
                            floatToStrf(rPercentage, ffFixed, 0, 1)); ✓
end;
```

(23)

```
//=============================================================================
```

// Q 3.1
(1) Display heading
(1) Initialise iTotalRob and iTotalCrime
(1) For loop stepping though months
(1) Initialise counter
(1) Add month number to string
(1) For stepping though characters in one line
(3) Test for uppercase characters B, D, H
(3) Add the lowercase version of B, D or H to string
(3) Test for R and add uppercase R to string and
(1) Increment counter
(1) Increment total crimes by counter
(1) Increments robberies
Per month:
(1) Add total robberies to string
(1) Display string
Outside loop per month:
(1) Calculate percentage
(2) Display percentage with label

// QUESTION 3.2

```
procedure TfrmQuest3.btnBClick(Sender: TObject);
var
  iLength, K, iCount, iFound, iMonth, iTheMonth  :integer;
  sString, sTheString :string;
  bFound :boolean;
  sCrime :string;
begin
  sString := 'b';
  bFound := false;          ✓
  iCount := 1;
  iFound := 1;

  while ((bFound = false) and (iFound <> 0)) do✓
  begin
   iMonth := 1; ✓
   while (bFound = false) and (iMonth <=12) do✓
   begin
    iFound := pos(Uppercase(sString), Uppercase(arrInfo[iMonth])); ✓
    if (iFound > 0) then✓
    begin
      bFound := true; ✓
      iTheMonth := iMonth; ✓
      sTheString := sString; ✓
     end
    else iMonth := iMonth + 1; ✓
   end;   // inner while
  if (bFound = true) then✓
    begin
      bFound := false; ✓
      sString := sString + 'b'; ✓
    end;
end;  // outer while


redOutput.Clear;
redOutput.Lines.Add('Crime: Burglaries'); ✓
redOutput.Lines.Add('The month with the most consecutive occurences: ' +
intToStr(iTheMonth)); ✓
redOutput.Lines.Add('The number of consecutive occurences during this month: '
+   intToStr(length(sTheString))); ✓


end;                                                                    (16)
//=============================================================================

end.
```

// Q 3.2
(1) Initialise variables
(1) Outer loop
(1) Initialise counter for months
(1) Inner loop
(1) Find the string of 'b's in string of characters
(1) if found
(1) set Boolean variable to true
(2) keep the month and string of 'b's
(1) else increment month
(3) below inner loop, if Boolean variable is true, add one 'b' character to the 'b'-string and reset Boolean variable for outer to false
(3) Display information

[39]

TOTAL: 120

ADDENDUM A

QUESTION 1: DELPHI – PROGRAMMING AND DATABASE

CENTRE NUMBER: ……	**EXAMINATION NUMBER: ……**

QUESTION 1: DELPHI – MARKING GRID

QUESTION	**ASPECT**	**MAX. MARKS**	**LEARNER'S MARKS**
1.1	SELECT SchoolName, Category, Balance, LastPayment (1)FROM tblSchools(1) ORDER BY Category(1), LastPayment(1)	**4**	
1.2	SELECT SchoolName, Balance, LastPayment FROM tblSchools (1) WHERE (Balance > 0) (1) AND (1) (#2011/11/25# (1) -LastPayment (1) > 60(1)) **ALTERNATELY:** *(DateDiff("d",[LastPayment],"2011/11/25")>60)*	**6**	
1.3	SELECT Officer, Count(1)(LogTime) AS [Peek Hours] FROM tblCallouts (1) WHERE (HOUR(1)(LogTime) Between(1) 15 and 24(1)) GROUP BY Officer(1)	**6**	
1.4	SELECT SchoolName, Reason , Count(LogDate) AS [Number of visits] (1) FROM tblSchools, tblCallouts (1) WHERE (tblSchools.SchoolNumber = tblCallouts.SchoolNumber)(1) AND (Year(LogDate) = 2011)(1) Group by SchoolName(1), Reason(1)	**6**	
1.5	SELECT Format(SUM(1)(Balance), "Currency") AS [Total amount], Format(1)(AVG(1)(Balance),"Currency") AS [Average amount], FROM tblSchools WHERE (Category = "S")(1)	**4**	
1.6	UPDATE tblSchools (1) SET(1) Balance = Balance - Instalments(1), LastPayment = #25/11/2011# (1) WHERE Category = "P" (1)	**5**	
1.7	DELETE(1) FROM tblCallouts(1) WHERE Year(1)(LogDate) = 2010(1)	**4**	
	TOTAL:	**35**	

ADDENDUM B

QUESTION 2: DELPHI – OBJECT-ORIENTED PROGRAMMING

CENTRE NUMBER:……	EXAMINATION NUMBER: ……

QUESTION 2: DELPHI – MARKING GRID

QUESTION	ASPECT	MAX. MARKS	LEARNER'S MARKS
2.1	**uHouseCheckXXXX_U.pas**		
2.1.1	**Attributes:** (1) Correct declare class (1) Attributes declared private (1) Correct attribute name and types	**3**	
2.1.2	**Constructor:** (1) Constructor declared correctly (1) Correct parameters (1) Values assigned correctly to attributes	**3**	
2.1.3	**isServiceNeeded:** (1) Method returns boolean (1) Receives character as parameter (2) Searches for character in string (1) Returns true or false accordingly	**5**	
2.1.4	**GetServicesForDay:** (1) Method receives an integer parameter (1) Returns a formatted string (1) Checks if service is required (2) on a particular day (1) Concatenates service onto return string (1) Checks for "None" condition	**7**	
2.1.5	**toString:** (1) Method called toString which returns a String (2) For loop for number of days (1) Joins day number and service for that day (1) Using getServicesForDay method (1) With a tab and newline	**6**	
2.1.6	**getTotalCost:** (1) Method returns a real number (1) Initialises a total variable to zero (1) Tests if a service is needed (2) Adds to cost for a service using correct formula	**5**	
2.1.7	**extendDays:** (1) Method declared correctly (1) Adds number of days (1) to days attribute	**3**	
2.2	**Question2XXXX_U.pas**		
2.2.1	**Object Declaration:** (1) THouseCheck object declared correctly (1) User enters a string and integer value (1) Calls constructor method (1) with correct parameters	**4**	
2.2.2	**Option A:** (1) Displays a suitable heading (1) Calls the toString method and displays result	**2**	
2.2.3	**Option B:** (1) Calls getTotalCost method (1) Displays result as a currency with suitable label	**2**	
2.2.4	**Option C:** (1) User enters day as integer (1) Calls getServicesForDay method (1) with parameter and (1) Displays the result	**4**	
2.2.5	**Option D:** (1) User enters days from as integer (1) Calls extendDays method with parameter	**2**	
	TOTAL:	**46**	

Copyright reserved/*Kopiereg voorbehou*

ADDENDUM C

QUESTION 3: DELPHI – PROGRAMMING

CENTRE NUMBER: ……	**EXAMINATION NUMBER: ……**

QUESTION 3: DELPHI – MARKING GRID

QUESTION	ASPECT	MAX. MARKS	LEARNER'S MARKS
3.1	(1) Display heading (1) Initialize iTotalRob and iTotalCrime (1) For loop stepping though months (1) Initialize counter (1) Add month number to string (1) For stepping though characters in one line (3) Test for uppercase characters B, D and H (3) Add the lowercase version of B, D or H to string (3) Test for R and add uppercase R to string and (1) Increment counter (1) Increment total crimes by counter (1) Increments robberies **Per month:** (1) Add total robberies to string (1) Display string **Outside loop per month:** (1) Calculate percentage (2) Display percentage with label	**23**	
3.2	(1) Initialise variables (1) Outer loop (1) Initialise counter for months (1) Inner loop (1) Find the string of 'b's in string of characters (1) if found (1) set Boolean variable to true (2) keep the month and string of 'b's (1) else increment month (3) below inner loop, if Boolean variable is true, add one 'b' character to the 'b'-string and reset Boolean variable for outer to false (3) Display information Or any other solution that supplies the correct output.	**16**	
	TOTAL:	**39**	

Copyright reserved/*Kopiereg voorbehou*