HILLCREST HIGH SCHOOL
INFORMATION TECHNOLOGY

GRADE 10 P1: PRACTICAL PAPER

TERM 4 NOVEMBER 2018

TIME: 3 HOURS TOTAL: 150 MARKS

EXAMINER: MR G. PEARL MODERATOR: MR G. ROSS

INSTRUCTIONS AND INFORMATION TO CANDIDATES

1. This paper consists of **FOUR** question(s).

2. Questions 1-4 are **compulsory**, all must be attempted and solutions saved.

3. Save your work regularly. Comment out errors when needed.
   Marks are awarded for commented out work.

4. Ensure that you have all files submitted and printed at the completion of the exam.

5. The duration of this test is **180** MINUTES. Due to the nature of the exam, you may not leave the examination room before the end of the exam session.

6. All data structures and built in functions can be used.

7. The following folder **[Grade10Exam2018]** have been given to you.

   Confirm that the data files are able to be opened once the examination is completed in the testing folder provided to you. Rename the Folder to your **ADMIN No**.

8. Marks will only be awarded for the printed out solution of your paper. In the event that the code is not printed, a review of the source code must also be possible. No marks will be awarded for any code that does not appear in both the saved code and the printed copy.

INFORMATION SHEET: COMMON DELPHI COMMANDS

Commands: Built In Type Conversion Functions	**Description / Example**
IntToStr(*int*)	*Converts from Integer (Wholes) to String*
FloatToStr(*float*)	*Converts from Floats (Fractions) to String*
BoolToStr(*Boolean*)	*Converts from Boolean (true/false) to String (-1, 0)*
DateToStr(*Date*)	*Converts from Date (any format) to String*
TimeToStr(*Time*)	*Converts from Time (any format) to String*
FloatToStrF(*float, format, i, d*)	*Converts from Floats (Fractions) to String Formatted*

Character Functions	
UPCASE(*char*)	**UPCASE**('x') → "X"
ORD(*char*)	**ORD**('A') → **65**
CHR(*number*)	**CHR**(65) → **'A'**
#9 Tab Space	*Adds a tab space character when used in TRichEdit or Textfiles*
#10 Line Feed	*Adds a line feed character when used in TRichEdit or Textfiles*
#13 Carriage Return	*Adds a carriage return character when used in TRichEdit or Textfiles*

String Functions	*var data: String; data := 'Examination';*
COPY(*string*, *from* ,*spaces*)	**Copy**(data,1,4) → "Exam"
POS(*needle*, *haystack*)	**Pos**('min',data) →**4**
DELETE(*string*, *from*, *spaces*)	**Delete**(data,1,); →ination
LENGTH(*string*)	**Length**(data) →**11**
UPPERCASE(*string*)	**Uppercase**(data) → 'EXAMINATION'

Maths Functions	
Random(*seedrange*)	**Random**(10) → A value from 0..9
Ceil(*float*)	**Ceil**(5.4) → **6**
Floor(*float*)	**Floor**(5.4) → **5**

Repetition Structures	**Conditional Structures**
var iLoop: Integer;	*conditions are mini-functions converted to Boolean statements*

For iLoop := 1 to 10 do
Begin …
End;
	IF (*condition***) THEN** …
	ELSE …

iLoop := 0;
WHILE (iLoop < 10) Do
Begin …
INC(iLoop);
End;
	CASE (*condition***) OF**
	condition **: …***handler;*
	condition **: …***handler;*
	ELSE …
	End;

iLoop := 0; **REPEAT …** **INC**(iLoop); **UNTIL** (iLoop = 10);	**Conditional Logic**: AND, OR, NOT **Set Operator Comparator: IN**['A'…'Z'] or **IN**[1..255]

Communication Dialogs	
ShowMessage(*message*);	**InputBox**(*title, label, default*)

QUESTION ONE

Scenario: Code Ninja
"*Ninja Warriors need to be able to perform various tasks with different GUI designed components in Delphi*"

In order to solve the question, a GUI form has been provided (*see* **APPENDEX A**).

PART A

1.1.1 On the event of **[1.1 Even]** button being pressed: [6]
Identify if the value entered into **edtNumber1** is an even or odd value, by displaying the appropriate message "YES" or "NO" using **lblEven**.

The program must convert the value into numerical data. Test to determine if the value is even or odd. If the value is even then "YES", if odd then "NO" must be displayed.

1.1.2 On the event of **[1.2 Best]** button being pressed: [6]
Identify the smallest value of the three components (**edtNumber1**, **seNumber2** and **seNumber3**). Display the smallest value using **lblMin**.

The program must allocate to seNumber3 a random value between 1-100. The smallest value of the three components should be determined and displayed.

1.1.3 On the event of **[1.3 Factor]** button being pressed: [8]
Identify if either values in **seNumber2** or **seNumber3** have the value in **edtNumber1** as their factor. Display the message "YES" and number of components it is a factor of, or "NO" using **lblFactor**.

The program must determine if both or either of the SpinEdit components have the value in edtNumber1 as a factor.
If **BOTH** are then display "YES – 2"
If **EITHER** is then display "YES – 1"
If **NONE** are then display "NO".

1.1.4 On the event of **[1.4Swap]** button being pressed: [3]
Swap the values in **seNumber2** and **seNumber3**.

TOTAL [23]

PART B

1.2.0 On the event of the Combo Box **[cboImage]** OnChange: [4]
Create a local String variable within the event handler called **filename**.
Store the selected text of the Combo Box into the local variable.
Load the associated picture file into the Image **imgPicture** component.

2.2.1 On the event of the **[2.1 Buy]** button being pressed: [6]
Create a global Real variable within the program called **fTOTAL**.
Extract the entered cost amount of an item.
Combine the item name, the '@' symbol and the cost of the item in a string.
Display the string in the **redout** TRichEdit.

2.2.2 On the event of the **[2.2 VAT and TOTAL]** button being pressed: [4]
Determine the NINJA VAT value if a **25%** VAT for NINJA's is charged.
Determine the new final total amount owed for all items.

Display the VAT value in the **redout** TRichEdit in the local currency.
Display the TOTAL value in the **redout** TRichEdit in the local currency.

2.2.3 On the event of the **[2.3 Ninja Coins]** button being pressed: [5]
Determine how many Ninja Coins represent the total amount.

Ninja's use THREE unique ninja coins as currency: Gopher, Geppy and Guppy.

Converted: R 1 =

- A **Gopher** is worth 50
- A **Geppy** is worth 5
- A **Guppy** is worth 1

Display the final number of coins needed per Rand, rounding up any cents.

IE: R 133.15 = R134 = **2** Gopher's, **6** Geppy's and **4** Guppy's

TOTAL [19]

QUESTION TOTAL [42]

QUESTION TWO

Scenario: Code Ninja

"*Ninja's must be as mentally sharp as Occam's razor. Keeping life simple with repetition is a powerful means for solving these coding riddles*"

In order to find the correct solution to each challenge or riddle, a button has been provided on the GUI with an associated TRichEdit component (*see* **APPENDEX B**).

PART A

2.1.1 On the event of **[2.1.1]** button being pressed: [4]

Using an appropriate looping statement, determine and display the square of all values from 1 to 10, each on a separate line in the **redoutA** component..

2.1.2 On the event of **[2.1.2]** button being pressed: [6]

Allow the user to enter a value to check using an InputBox.

Check all values from 1 to 20 to determine if they are factors of the value entered by the user. In the event that they are a factor, display the value in the **redoutA** component.

2.1.3 On the event of **[2.1.3]** button being pressed: [6]

Using appropriate looping statements, create and display the grid below:

1A	**1B**	**1C**	**1D**
2A	**2B**	**2C**	**2D**
3A	**3B**	**3C**	**3D**

2.1.4 On the event of **[2.1.4]** button being pressed: [8]

Allow the user to enter a value to check using an InputBox if it is a Prime.

Check if the user has entered a Prime value or not. In the event that the value entered is a Prime, display the value and the message that it is a Prime.

TOTAL [24]

PART B

2.2.1 On the event of the **[2.2.1]** button being pressed: [6]
Solve the following Riddle using the appropriate looping statements:

"*With sevens and threes I am found, my value is bigger than 50.*"

Calculate and show what the first possible value is using the **redoutB** component.

2.2.2 On the event of the **[2.2.2]** button being pressed: [6]
Solve the following Riddle using the appropriate looping statements:

"*In two years I know, I'll be twice as old as five years ago, said Tom.*"

Calculate and show how old Tom is using the **redoutB** component.

2.2.3 On the event of the **[2.2.3]** button being pressed: [6]
Solve the following Riddle using the appropriate looping statements:

"*A Square room is equal in sides. The length of the square room is greater than 5, once rounded*".

Calculate and show, the minimum area for the square room using the **redoutB** component.

2.2.4 On the event of the **[2.2.4]** button being pressed: [8]
A palindromic number remains the same when its digits are reversed. Write the event handler's code so that it allows the Ninja user to enter a value for C and generates a simple palindromic number.

The program must generate a simple palindromic numbers using the following pattern
12… C …21

IE: If C is 3 then 12321
If C is 4 then 1234321

TOTAL [26]
QUESTION TOTAL [50]

QUESTION THREE

Scenario: Code Ninja
"*No secret can escape the ears of a Ninja, he hears and understands all*"

A Ninja must be able to decrypt secret messages quickly using the provided GUI.

PART A

3.1 On the event of **[3.1.1]** button being pressed: [19]

Using the two (Part A and Part B) edit components, allow the user to enter and combine these parts to form a single message.
IE: "**ZULU WARRIORS**"[PART A] "ATTACK"[PART B]

If both parts are the same, indicate this by displaying the message: "Both parts are the same".

Use the Caesar cipher, shifting the characters by one value in the Alphabet to encrypt the message. Do not apply UpperCase.

A	B	…	Y	Z		a	B	…	Y	Z
B	C	…	Z	A		b	c	…	z	A

Display the final result in the **edtCode** and **redOutA** components.

[19]

TOTAL

PART B

3.1 On the event of the **[3.2.1]** button being pressed: [3]
Determine the number of characters that have been entered into the **edtMessage** (TEdit) component.

Test using the following:

- "Zulu Warriors Attack" = 20
- "Ninja@gmail.com" = 15

3.2 On the event of the **[3.2.2]** button being pressed: [4]
Determine where the '@' character occurs within the message that has been entered into the **edtMessage** (TEdit) component.

Test using the following:

- "Zulu Warriors Attack" = 0
- "Ninja@gmail.com" = 6

3.3 On the event of the **[3.2.3]** button being pressed: [5]
Extract only the first three characters occurring within the message that has been entered into the **edtMessage** (TEdit) component.

Test using the following:

- "Zulu Warriors Attack" = Zul
- "Ninja@gmail.com" = Nin

3.4 On the event of the **[3.2.4]** button being pressed: [4]
Extract all the characters except the first three characters occurring within the message that has been entered into the **edtMessage** (TEdit) component.

IE: remove the unwanted first three characters.

Test using the following:

- "Zulu Warriors Attack" = u Warriors Attack
- "Ninja@gmail.com" = ja@gmail.com

TOTAL [16]

QUESTION TOTAL [35]

QUESTION FOUR

Scenario: Code Ninja

"The Chinese Emperor has decided to reward you with one of his daughters to marry."

However, there is a twist; you get **ONE** second to see them and need to choose the most beautiful daughter in order to obtain the kingdom.

PART A

In order to solve the problem a GUI and list of princesses (names and scores) have been provided.

On the event of **[4.1.1 Best in List]** button being pressed: [8]

Identify the most beautiful princess. First select the first listed princess's score and then search all the listed scores for the best scored princess.

The princess with the best score is to be selected. In the unlikely event of two or more princesses having the same score, the first listed princess must be highlighted in the score list (found on the left).

On the event of **[4.1.2 Above 1000 pts]** button being pressed: [4]

Count the number of princesses that scored above 1000 points and display the amount in the label provided below.

TOTAL [12]

CODE BLOCK
`begin`
`end;`
`for iLoop := 0 to princesses do`
`if iScore > 1000 then`
`iMaxIndex := 0;`
`iMaxValue :=`
`INC(iCount);`
`IntToStr(iCount);`
`iScore :=`
`lblAbove1000.Caption :=`
`StrToInt(lstScore.Items[0]);`
`StrToInt(lstScore.Items[iLoop]);`

Scenario: Code Ninja

"The Chinese Emperor has declared that your Ninja name will be remembered in the history of China as one of the greatest."

As Ninja's enter into the scroll of History, the compiling of their deeds is measured and kept in the sacred history of China.

PART B

In order to solve the problem a GUI and list of past Ninja Warriors (scores and names) have been provided.

On the event of **[4.2.1**  **]** button being pressed: [6]

Allow the user to enter into the two TEdit components the score and name of the Ninja.

Add the combined score and name, separated by the '|' symbol to the list. Sort the list and save the list to the text-file.

On the event of **[4.2.2**  **]** button being pressed: [5]

Allow the user to select prior on the list the history item that they want to delete. Check that an item has been selected. Confirm with the user that they want to delete the selected item.

Delete the item from the list and save the list to the text-file.

TOTAL [11]

CODE BLOCK
`+ ' | ' + itemName`
`if (answer = mrOK) then`
`if iSelect > -1 then`
`iSelect := lstHistory.ItemIndex;`
`item := lstHistory.Items[iSelect];`
`itemName :=`
`itemScore := edtScore.Text;`
`lstHistory.Items.Add(item);`
`lstHistory.Items.Delete(iSelect);`
`lstHistory.Items.SaveToFile('HighScores.txt');`
`lstHistory.Sorted := True;`

CODE AS PROVIDED FOR **PART A**

```
{ CODE PROVIDED: DO NOT EDIT ANY CODE ABOVE }

procedure TForm1.btnBestInListClick(Sender: TObject);
var
  iLoop, iMaxValue, iScore, iMaxIndex: Integer;
begin
  {A1}
  {A2} {A3}

  {A4}
  begin
    {A5} {A6}

    if (iScore > iMaxValue) then
    {A7}
      iMaxValue := iScore;
      iMaxIndex := iLoop;
    {A8}
  end;

  lstScore.ItemIndex := iMaxIndex;
end;

procedure TForm1.btnAbove1000Click(Sender: TObject);
var
  iLoop, iScore, iCount: Integer;
begin
  iCount := 0;

  {A4}
  begin
    {A5}{A6}

    {A9}
       {A10}
  end;

  {A11} {A12}
end;

end.
```

CODE BLOCK
begin
end;
for iLoop := 0 to princesses do
if iScore > 1000 then
iMaxIndex := 0;
iMaxValue :=
INC(iCount);
IntToStr(iCount);
iScore :=
lblAbove1000.Caption :=
StrToInt(lstScore.Items[0]);
StrToInt(lstScore.Items[iLoop]);

CODE AS PROVIDED FOR **PART B**

```
procedure TForm1.ToolButton1Click(Sender: TObject);
var
  itemScore: String;
  itemName: String;
  item: String;
begin
  //
  {A1}
  {A2} edtName.Text;
  item := itemScore {A3};
  {A4}
  {A5}
  {A6}
end;

procedure TForm1.ToolButton2Click(Sender: TObject);
var
  item: String;
  iSelect: Integer;
  answer: Integer;
begin
  //
  {A7}
  {A8}
  begin
    {A9}
    answer := MessageDlg('Confirm deleting of ' + item, mtConfirmation,
      mbOKCancel, 0);
    {A10}
      {A11}
    {A6}
  end;
end;
```

CODE BLOCK
+ ' | ' + itemName
if (answer = mrOK) then
if iSelect > -1 then
iSelect := lstHistory.ItemIndex;
item := lstHistory.Items[iSelect];
itemName :=
itemScore := edtScore.Text;
lstHistory.Items.Add(item);
lstHistory.Items.Delete(iSelect);
lstHistory.Items.SaveToFile('HighScores.txt');
lstHistory.Sorted := True;