Province of the
EASTERN CAPE
EDUCATION

NATIONAL
SENIOR CERTIFICATE

GRADE 11

NOVEMBER 2019

INFORMATION TECHNOLOGY P1

MARKS: 150

TIME: 3 hours

IINFTE1

This question paper consists of 13 pages.

INSTRUCTIONS AND INFORMATION

1. This question paper consists of FOUR questions. Candidates must answer ALL four questions.
2. The duration of this examination is three hours. Because of the nature of this examination it is important to note that you will not be permitted to leave the examination room before the end of the examination session.
3. Answer only what is asked in each question. For example, if the question does not ask for data validation, then no marks will be awarded for data validation.
4. Your programs must be coded in such a way that they will work with any data and not just the sample data supplied or any data extracts that appear in this question paper.
5. Routines such as search, sort and selection must be developed from first principles. You may NOT use the built-in features of a programming language for any of these routines.
6. Save your work regularly. Ensure that all files can be read.
7. The files that you need to complete this question paper have been given to you. The files are provided in the form of password-protected executable files.

   Do the following:

   - Double click on the password-protected executable file.
   - Click on the extract button.
   - Enter the following password: **Gr11&ScL19**

   Once extracted, the following list of files will be available in the folder **DataNov2019**:

   Question 1:

   Question1_u.pas
   Question1_u.dfm
   Question1_p.dpr
   Question1_p.res

   Question 2:

   Question2_u.pas
   Question2_u.dfm
   Question2_p.dpr
   Question2_p.res
   dbConnection_u.pas
   TechShop.mdb
   TechShopBackUp.mdb

   Question 3:

   Question3_u.pas
   Question3_u.dfm
   Question3_p.dpr
   Question3_p.res

   Question 4:

   Question4_u.pas
   Question4_u.dfm
   Question4_p.dpr
   Question4_p.res
   sales.txt

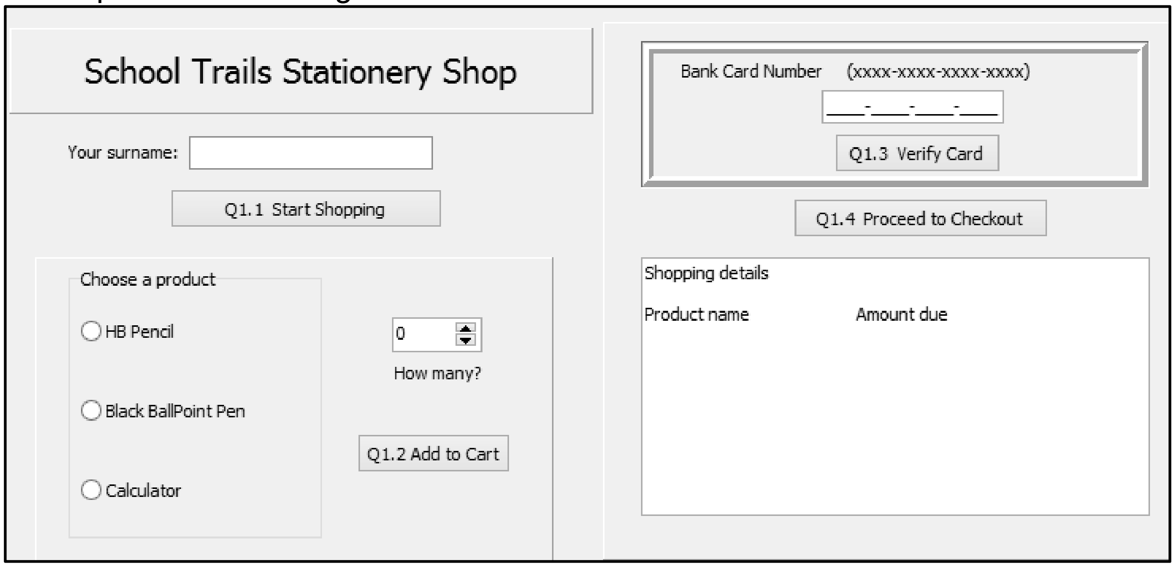
QUESTION 1: GENERAL PROGRAMMING SKILLS

You have been asked by the manager of your school's stationery shop to assist with developing an online shopping alternative.

Do the following:

- Open the incomplete program in the **Question 1** folder.
- Enter your full name as a comment in the first line of the **Question1_u.pas** file.
- Compile and execute the program. The program currently has no functionality.
- Follow the instructions to complete the code for QUESTION 1.1 to QUESTION 1.4.

An example of the GUI is given below:

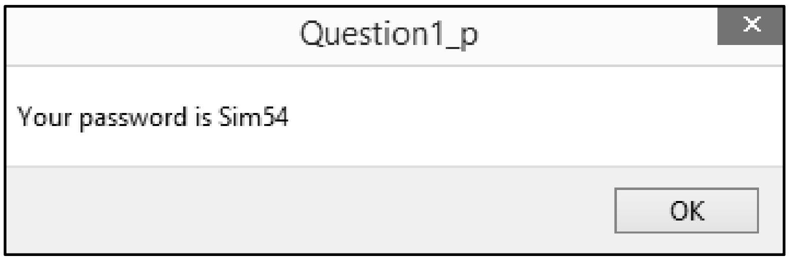


1.1 **Button [Q1.1 – Start Shopping]**

Customers are required to have a password to start shopping. They will enter their <u>surname only</u> in the edit box named **edtname** and you must write code to generate a password consisting of the first **three** characters of the surname joined to a random number from 1 to 1034 (both included).

You must write code to show a message to display the password.

Example of output if 'Simons' is entered as the surname:



```
Your password is Sim54
```

(6)

1.2 **Button [Q1.2 – Add to Cart]**

The user will choose a product from the radio group component named **rpgChoice** and then they will enter the quantity that they require for that product in the spinedit named **sedQty**.

The quantity for the product chosen must be multiplied by the unit price for that product to obtain the amount due.

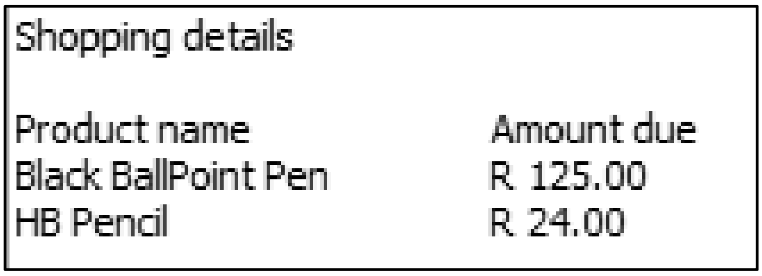
Unit prices are provided as constants in the program:
Pencil = R12,00, Pen = R25,00, Calculator = R154,00

Display the results in the richedit component, **redInvoice**, in neat columns as shown in the example of output below.

NOTE: Code to provide spacing and to display the headings has been provided.

Customers will choose more than one product and therefore a total amount due must be calculated using the calculated amount due for a chosen product. This total amount will be required in QUESTION 1.4.
(**Hint**: Use the global variable **rGrandTotal** provided).

Example of output if 5 black ballpoint pens and 2 HB pencils are ordered:



```
Shopping details

Product name              Amount due
Black BallPoint Pen       R 125.00
HB Pencil                 R 24.00
```

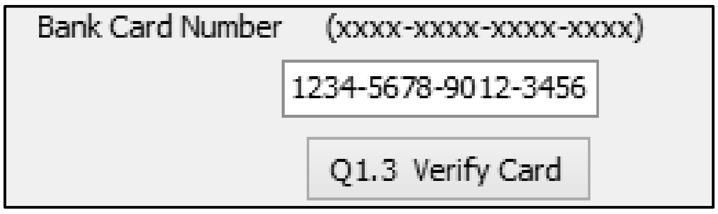
(13)

1.3 **Button [Q1.3 – Verify Card]**

Code must be written to ensure that the bank card number entered by the customer is valid.
The edit component named **edtCard** has an editmask applied which will ensure that the character **'-'** is placed correctly between 4 sets of 4 digits.

A card is valid if one of the numbers 0,1,2,3,4,5,6,7,8 and 9 are entered for each set of 4 digits.

Example of a bank card number:

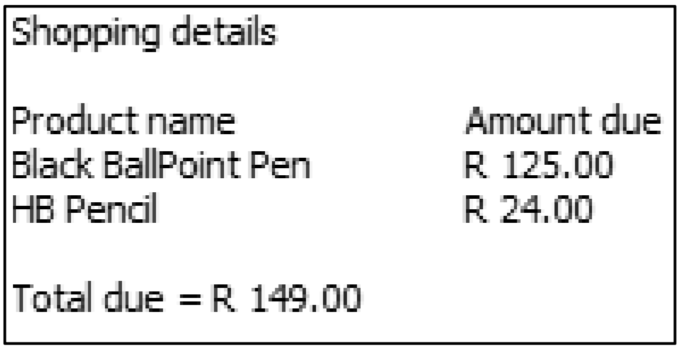


You must write code to test if there are other characters which are not numbers from 0 to 9 in the card number.
If the card is valid, then set the font style of the button named **btnQ1_4** to bold font.
If the card number is invalid, then display a suitable message. (15)

1.4 **Button [Q1.4 – Proceed to Checkout]**

The total due for all products for a customer must be displayed in the richedit, **redInvoice**, formatted to currency and two decimal places.

Example of output if 5 pens and 2 pencils are ordered:



```
Shopping details

Product name            Amount due
Black BallPoint Pen     R 125.00
HB Pencil               R 24.00

Total due = R 149.00
```

(4)

- Enter your name and surname as a comment in the first line of the program file.
- Save your program.
- A printout of the code may be required.

[38]

QUESTION 2: DATABASE MANIPULATION

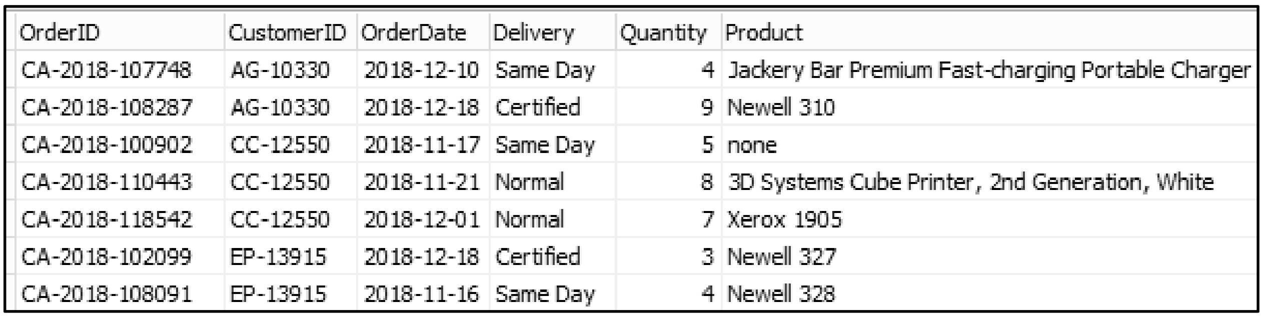
This question consists of sub-questions 2.1 to 2.4. The following important notes are applicable to all questions:

- You are NOT allowed to modify or add to the supplied data or supplied programming code.
- Good programming techniques from first principles only, must be applied.
- NO marks will be assigned for hardcoding. Use control structures and variables where necessary.
- **NO FILTERS MAY BE USED.**

Your school's stationery shop has extended its business to include technical sales, including cellphones and various office accessories. A database has been designed for this purpose and all orders are recorded in one table.

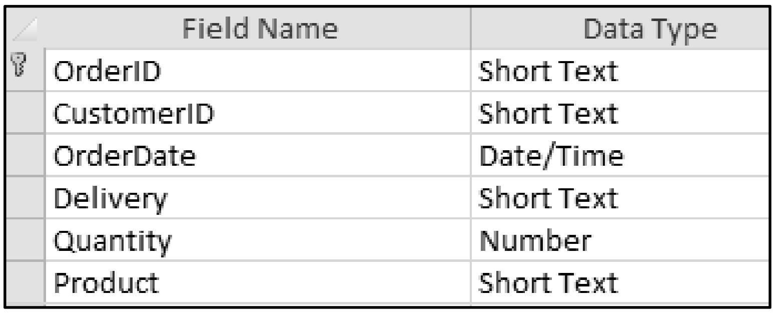
A table named Orders has been supplied in the database named **TechShop.mdb.**

A sample of records in this table is displayed below:



OrderID	CustomerID	OrderDate	Delivery	Quantity	Product
CA-2018-107748	AG-10330	2018-12-10	Same Day	4	Jackery Bar Premium Fast-charging Portable Charger
CA-2018-108287	AG-10330	2018-12-18	Certified	9	Newell 310
CA-2018-100902	CC-12550	2018-11-17	Same Day	5	none
CA-2018-110443	CC-12550	2018-11-21	Normal	8	3D Systems Cube Printer, 2nd Generation, White
CA-2018-118542	CC-12550	2018-12-01	Normal	7	Xerox 1905
CA-2018-102099	EP-13915	2018-12-18	Certified	3	Newell 327
CA-2018-108091	EP-13915	2018-11-16	Same Day	4	Newell 328

The design view of the table named Orders, which includes the field data types, is displayed below:



Field Name	Data Type
OrderID	Short Text
CustomerID	Short Text
OrderDate	Date/Time
Delivery	Short Text
Quantity	Number
Product	Short Text

NOTE:

- Connection code has been provided.
- When the **Restore Database** button is clicked, the data in the database will be restored to the original data.
- The name of the table to be used in your code must be **tblOrder,** which is a TADOTable object connected to the database.

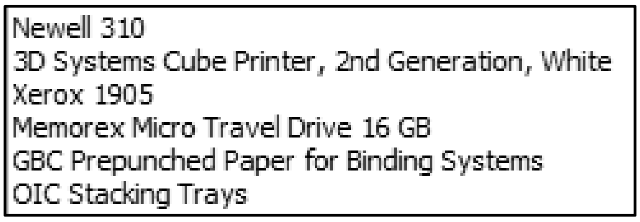
Do the following:

- Compile and execute the program in the **Question 2** folder. The program currently has limited functionality.
- Enter your full name as a comment in the first line of the **Question2_u.pas** file.
- Complete the code for each question as described in Question 2.

2.1 **Button [Q2.1 – Quantity over 5]**

Display the names of all products where the quantity ordered is greater than 5. Display the results in the richedit named **redDisplay.**

Example of output:



```
Newell 310
3D Systems Cube Printer, 2nd Generation, White
Xerox 1905
Memorex Micro Travel Drive 16 GB
GBC Prepunched Paper for Binding Systems
OIC Stacking Trays
```

(6)

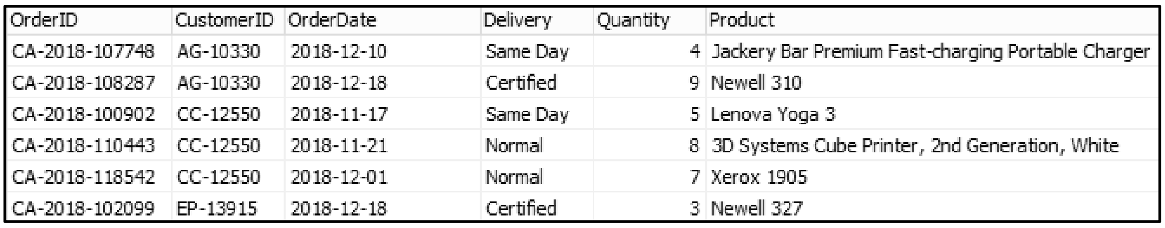
2.2 **Button [Q2.2 – Change ONE Product]**

There exists one order where the **Product** name is listed as '*none*' in the database table. This was a result of an error when data entry occurred.

You must write code to change that **Product** name from 'none' to 'Lenova Yoga 3'

NOTE: There is only ONE order where this error was made.

Example of output:



OrderID	CustomerID	OrderDate	Delivery	Quantity	Product
CA-2018-107748	AG-10330	2018-12-10	Same Day	4	Jackery Bar Premium Fast-charging Portable Charger
CA-2018-108287	AG-10330	2018-12-18	Certified	9	Newell 310
CA-2018-100902	CC-12550	2018-11-17	Same Day	5	Lenova Yoga 3
CA-2018-110443	CC-12550	2018-11-21	Normal	8	3D Systems Cube Printer, 2nd Generation, White
CA-2018-118542	CC-12550	2018-12-01	Normal	7	Xerox 1905
CA-2018-102099	EP-13915	2018-12-18	Certified	3	Newell 327

(9)

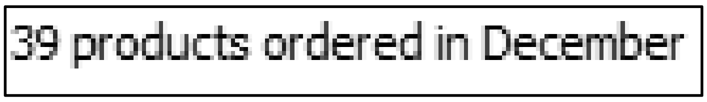
2.3 **Button [Q2.3 – December orders]**

Write code to display the total of the quantities of products for orders that were placed in the month of December.

All dates are represented as "yyyy-mm-dd".

The total must be displayed in the richedit named **redDisplay**.

Example of output:



```
39 products ordered in December
```

(10)

2.4 **Button [Q2.4 – Delete Customer]**

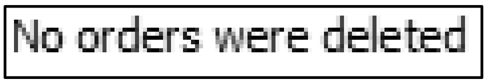
The user will choose a Customer ID from the combo box named *cmbcustomer*.

You must write code to DELETE all records from the database table which contain the **CustomerID** chosen by the user AND where the type of **Delivery** is listed as Normal.

Before deleting a record, you must display the **OrderID** of the record to be deleted in the RichEdit named *redDisplay*.

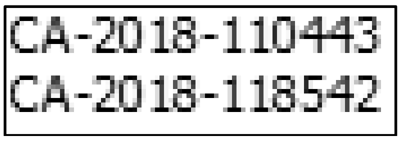
If no record is deleted then display a message in the richedit: "*No records were deleted".*

Example of the display in *redDisplay* if Customer ID **AG-10330** is selected:



```
No orders were deleted
```

Example of the display in *redDisplay* if Customer ID **CC-12550** is selected:



```
CA-2018-110443
CA-2018-118542
```

(13)

- Enter your name and surname as a comment in the first line of the program file.
- Save your program.
- A printout of the code may be required.

[38]

QUESTION 3: ARRAYS AND SUBPROGRAMS

The stationery shop would like to display reports of stock quantities.

The program provided in the QUESTION 3 folder includes two global arrays named ***arrItem*** and ***arrQty***.

ArrItem contains 10 names of items including the quantities which must be added to the stock numbers of each item.

The array named *ArrQty* contains the old quantities in stock of each item and each index is related to the corresponding index of *arrItem.*

For example, the first index of *arrQty* contains the current amount in stock, which is three, of red ballpoint pens.

3.1 Write a procedure named **SortandDisplay** which will sort both arrays named *arrItem* and *arrQty*.

The arrays must be sorted in alphabetical order of the array named *arrItem*.

This method must then display both arrays <u>in neat columns</u> in the richedit named *redOutput*. (*arrItem followed by arrQty*) (15)

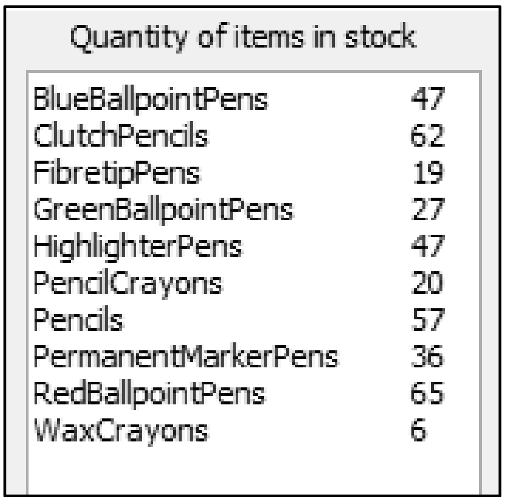
3.2 **OnCreate event handler of frmQuestion3**

The array named *arrItem* contains the item name as well as new quantities to be added to the stock for each item.

Write code to <u>remove</u> the quantities in the indices of *arrItem* and <u>add</u> these numbers to the quantities in stock in the array named *arrQty*.

Call the method named **SortandDisplay** to display the arrays in the richedit.

Example of output:



```
Quantity of items in stock

BlueBallpointPens        47
ClutchPencils            62
FibretipPens             19
GreenBallpointPens       27
HighlighterPens          47
PencilCrayons            20
Pencils                  57
PermanentMarkerPens      36
RedBallpointPens         65
WaxCrayons               6
```

(8)

3.3 Write a function named **GetNewQuantity** which will receive TWO integer parameters and return an integer value.

One parameter is an integer representing the quantity in stock.
The second parameter is an integer representing the percentage with which to decrease the stock quantity.

The function must return an integer which will be the new decreased stock value, based on the following rule:

A quantity in stock cannot be decreased by a fraction of a number. You must round off to the nearest lowest whole number.

(8)

	INPUT		**PROCESSING**		**OUTPUT**
	Stock Quantity	**Percentage Decrease**	**Result after calculation**	**Quantity that must be subtracted**	**Result that function must return**
Example 1	35	10	3.5	3	32
Example 2	42	4	1.68	1	41
Example 3	20	12	2.4	2	18

3.4 **Button [Q3.4 – Stock Decrease]**

After a month of sales, the stock numbers will decrease.

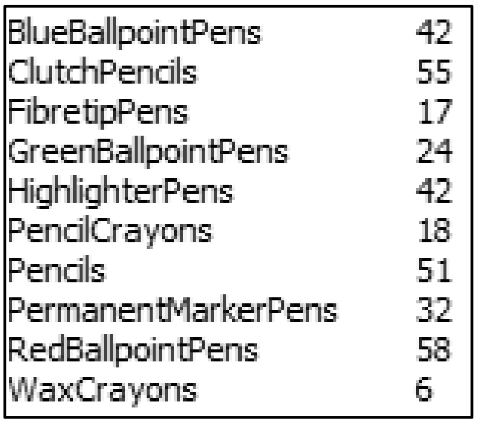
Write code for the user to enter an integer into an input component to represent the percentage with which to decrease the stock.

The quantities in stock of all items must then be reduced by this percentage, using the global array named **arrQty** and the method named **GetNewQuantity** which you wrote in QUESTION 3.3.

(The array named **arrQty** must be updated with the new values calculated using the method named **GetNewQuantity**.)

Call the method **SortandDisplay** which you wrote in QUESTION 3.1 to display the results in the rich edit, to test your solution.

Example of output if stock quantities are reduced by 12 percent:



```
BlueBallpointPens        42
ClutchPencils            55
FibretipPens             17
GreenBallpointPens       24
HighlighterPens          42
PencilCrayons            18
Pencils                  51
PermanentMarkerPens      32
RedBallpointPens         58
WaxCrayons               6
```

(7)

- Enter your name and surname as a comment in the first line of the program file.
- Save your program.
- A printout of the code may be required.

[38]

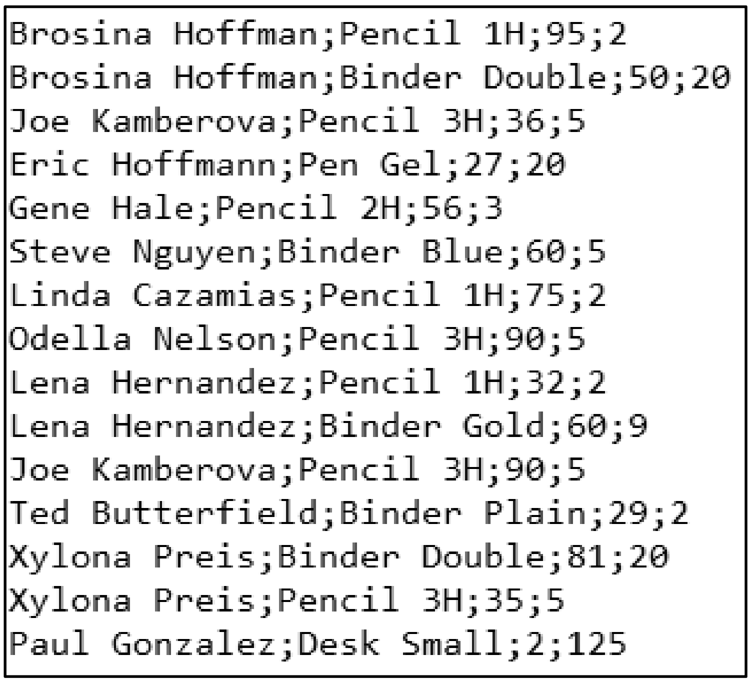
QUESTION 4: TEXT FILE MANIPULATION

A list of sales details for the stationery company is stored in a text file named **sales.txt**.

The layout of one line in the text file is as follows:

customer name;item sold;quantity sold;unit price of item sold

NOTE: Each entry on one line in the text file is separated by a semicolon (;)



```
Brosina Hoffman;Pencil 1H;95;2
Brosina Hoffman;Binder Double;50;20
Joe Kamberova;Pencil 3H;36;5
Eric Hoffmann;Pen Gel;27;20
Gene Hale;Pencil 2H;56;3
Steve Nguyen;Binder Blue;60;5
Linda Cazamias;Pencil 1H;75;2
Odella Nelson;Pencil 3H;90;5
Lena Hernandez;Pencil 1H;32;2
Lena Hernandez;Binder Gold;60;9
Joe Kamberova;Pencil 3H;90;5
Ted Butterfield;Binder Plain;29;2
Xylona Preis;Binder Double;81;20
Xylona Preis;Pencil 3H;35;5
Paul Gonzalez;Desk Small;2;125
```

Button [Q4 – Analyse sales]

A user will choose an item name from the combobox named *cmbitem.*

You must write code to:

- Test if the text file named **sales.txt** exists and exit the method if it does not exist.
- Extract the item name from the combobox
- Read a line from the text file.
- Test if the item chosen from the combobox is a part of / exists in the line from the text file.
- If there is a match, then you must:
  - Extract the customer's name and item name
  - Extract the quantity sold and the unit price from the text file line.
  - Calculate the cost to the customer for the item sold (unit price multiplied by the quantity sold).
  - Calculate the total of all costs. (This will be used later)
  - Display the name of the customer and the cost to that customer in the richedit named *redDisplay* provided.

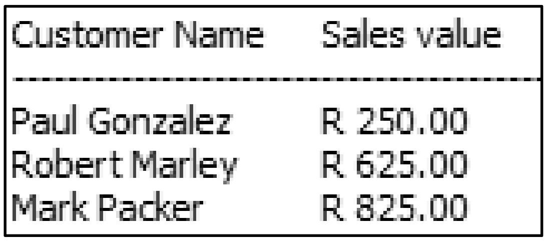
Write code to test if a text file exists which is named with the item name extracted from the combobox.

- If the text file does exist, then you must write code to prepare the text file to have lines added to it.
- If the text file does not exist, then you must write code to create a text file using the item name from the combobox as the new text file name.

You must then write code to add a line to the text file using the following format:

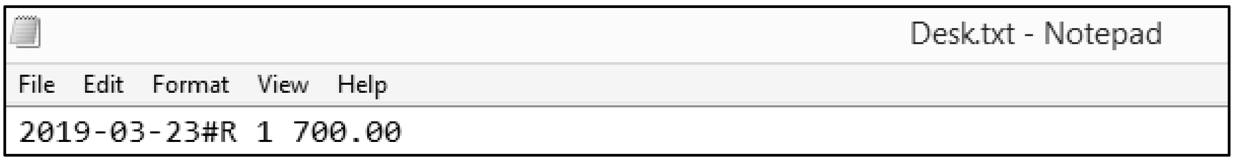
Today's date#total of all costs for the item

Example of output in the richedit if **Desk** is chosen from the combobox:



```
Customer Name    Sales value
------------------------------------
Paul Gonzalez    R 250.00
Robert Marley    R 625.00
Mark Packer      R 825.00
```

Example of output to the text file named **Desk.txt**:



```
2019-03-23#R 1 700.00
```

- Enter your name and surname as a comment in the first line of the program file.
- Save your program.
- A printout of the code may be required.

[36]

TOTAL: 150