



# basic education

---

Department:  
Basic Education  
**REPUBLIC OF SOUTH AFRICA**

**NATIONAL  
SENIOR CERTIFICATE**

**GRADE 12**

**INFORMATION TECHNOLOGY P1**

**NOVEMBER 2017**

**MARKS: 150**

**TIME: 3 hours**

**This question paper consists of 18 pages.**

**INSTRUCTIONS AND INFORMATION**

1. This question paper is divided into THREE sections. Candidates must answer ALL the questions in ALL THREE sections.
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. This question paper is set with programming terms that are specific to Delphi programming language.
4. Make sure that you answer the questions according to the specifications that are given in each question. Marks will be awarded according to the set requirements.
5. 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.
6. 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 the question paper.
7. Routines, such as search, sort and selection, must be developed from first principles. You may NOT use the built-in features of Delphi for any of these routines.
8. All data structures must be defined by you, the programmer, unless the data structures are supplied.
9. You must save your work regularly on the disk/CD/DVD/flash disk you have been given, or on the disk space allocated to you for this examination session.
10. Make sure that your examination number appears as a comment in every program that you code, as well as on every event indicated.
11. If required, print the programming code of all the programs/classes that you completed. You will be given half an hour printing time after the examination session.
12. At the end of this examination session you must hand in a disk/CD/DVD/flash disk with all your work saved on it OR you must make sure that all your work has been saved on the disk space allocated to you for this examination session. Make sure that all files can be read.

13. The files that you need to complete this question paper have been given to you on the disk/CD/DVD/flash disk or on the disk space allocated to you. The files are provided in the form of password-protected executable files.

**NOTE:**

Candidates must use the file **DataENGNov2017.exe**.

Do the following:

- Double click on the password-protected executable file.
- Click on the extract button.
- Enter the following password: **On\$LiNe17**

Once extracted, the following list of files will be available in the folder **DataENGNov2017**:

**SUPPLIED FILES****Question 1:**

Pict1.png  
Pict2.png  
Pict3.png  
Pict4.png  
Pict5.png  
Pict6.png  
Pict7.png  
Pict8.png  
Pict9.png  
Pict10.png  
Question1\_P.dpr  
Question1\_P.res  
Question1\_U.dfm  
Question1\_U.pas

**Question 2:**

DCertificate\_U.pas  
DigitalCertificates.txt  
Question2\_P.dpr  
Question2\_P.res  
Question2\_U.dfm  
Question2\_U.pas

**Question 3:**

Question3\_P.dpr  
Question3\_P.res  
Question3\_U.dfm  
Question3\_U.pas

**SECTION A****QUESTION 1: GENERAL PROGRAMMING SKILLS**

Do the following:

- Open the incomplete program in the **Question 1** folder.
- Enter your examination number as a comment in the first line of the **Question1\_U.pas** file.
- Compile and execute the program. The program currently has no functionality.
- The program contains FIVE tab sheets with different unrelated questions.
- Follow the instructions below to complete the code for EACH section of QUESTION 1, as described in QUESTION 1.1 to QUESTION 1.5.

**1.1 Tab sheet [Question 1.1]**

Write code in the **OnCreate** event handler of the form to do the following:

- Display the sentence 'IT is FUN!' on the panel **pnIQ1\_1** provided.
- Set the background colour of the panel to lime.
- Set the font size of the text to 15.

Example of output when the program is executed:



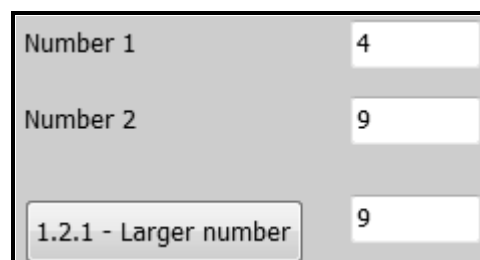
(3)

**1.2 Tab sheet [Question 1.2]****1.2.1 Button [1.2.1 – Larger number]**

Write code to do the following:

- Extract number 1 and number 2 entered by the user from the edit boxes provided.
- Determine and display the larger number in the edit box **edtQ1\_2\_1**.
- If the entered numbers are the same, display the word 'Equal' in the edit box **edtQ1\_2\_1**.

Example of output for input values 4 and 9:



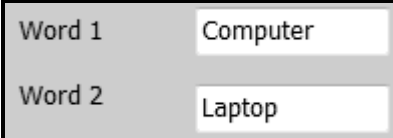
(4)

### 1.2.2 Button [1.2.2 – Swap words]

Write code to do the following:

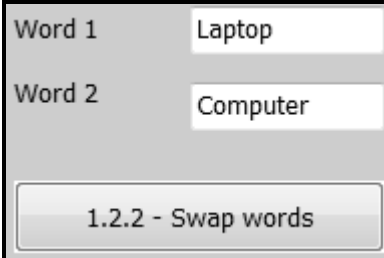
- Extract word 1 and word 2 entered by the user from the edit boxes provided.
- Use the variables provided to store these values.
- Swap the contents of the two variables.
- Display the words in the edit boxes after swapping.

Example of input:



A screenshot of a form with two text input fields. The first field is labeled 'Word 1' and contains the text 'Computer'. The second field is labeled 'Word 2' and contains the text 'Laptop'.

Example of output:



A screenshot of a form showing the result of the swap. The 'Word 1' field now contains 'Laptop' and the 'Word 2' field contains 'Computer'. Below the fields is a button labeled '1.2.2 - Swap words'.

(5)

### 1.3 Tab sheet [Question 1.3]

Best Buy Bakery is a small business that bakes cakes for the community.

The combo box **cmbNumCakes** is populated with numbers from 1 to 10, which are used to select the number of cakes to be ordered by the customer. There are ten picture files in the folder to represent the number of cakes ordered.

An image component, **imgCakePic**, is used to load a picture that shows the number of cakes to be ordered. Example: If the number of cakes selected is three, the name of the picture file will be **Pict3.png**.

#### 1.3.1 [Combo box]

The price of a cake is saved in a constant variable called **PRICE**, which contains the value 159.50.

Write code for the combo box to do the following:

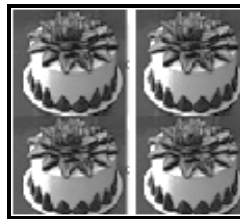
- Extract the number of cakes selected from the combo box.
- Display the picture that represents the number of cakes selected.

- Use the selected number of cakes and the value of the constant variable (PRICE) to calculate the cost of the number of selected cakes. Display the cost using currency with TWO decimal places.

Example of input and output if four cakes are selected:

Number of cakes	4	Cost	R638.00
-----------------	---	------	---------

Example of output when the picture file **Pict4.png** is loaded:



(5)

### 1.3.2 Button [1.3.2 – Calculate the amount of sugar]

A single cake requires 375 grams of sugar.  
Sugar is packed in quantities of 1 kg only.

Write code to do the following:

- Calculate the amount of sugar in grams that is required for the number of cakes selected.
- Determine the number of 1 kg packets of sugar that must be purchased.

**NOTE:** 1 000 grams = 1 kilogram

- Display the amount of sugar required in grams, as well as the number of 1 kg packets of sugar to be purchased.

Example of output for four cakes selected:

Sugar in grams	1500	1 kg packet(s) of sugar	2
----------------	------	-------------------------	---

(5)

## 1.4 Tab sheet [Question 1.4]

### 1.4.1 Radio group [Type of user]

A password is required for the owner and the staff at the bakery to use the system. The panel **pnlQ1\_4**, where the password must be entered, is not visible to the user.

Write code to display the panel (**pnlQ1\_4**) if Owner or Staff is selected from the radio group and to hide the panel if Customer is selected.

Example if Owner was selected from the radio group. The panel **pnIQ1\_4** will be visible.



(4)

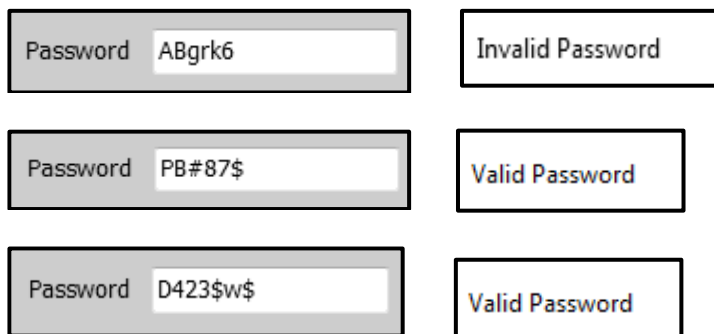
1.4.2 **Button [1.4.2 – Validate password]**

Write code to do the following:

- Extract the password from the edit box.
- Determine whether the password is valid or not. A valid password must satisfy the following criteria:
  - It must consist of at least six characters.
  - The first character must be a capital letter.
  - The password must contain two or more special characters. The special characters must be from the following list:  
\$, @, #, &

Characters may be repeated. (See examples below.)
- Use a dialog box to display a message indicating whether the password is valid or not.
- Enable button **btnQ1\_4\_3** if a valid password was entered.
- Clear the Password edit box if the password is invalid.

Examples of input and output of valid and invalid passwords:



(11)

1.4.3 **Button [1.4.3 – Encrypt password]**

Write code to do the following:

- Change the first letter of the valid password to the next alphabetical letter, for example if the first letter is 'A', the letter must become 'B'. If the first letter is 'B', the letter must become 'C', and so on. If the first letter is 'Z', the letter must become 'A'.
- Display the encrypted password in the password edit box.

Example of output for the valid password PB#87\$:

A screenshot of a password input field. The text 'Password' is on the left, and the input box contains 'QB#87\$'.

(5)

1.5 **Tab sheet [Question 1.5]**

1.5.1 **Button [1.5.1 – Perfect square]**

A perfect square is the result of an integer value multiplied by itself.

Write code to enter an integer value using an InputBox. Display a message in the output area **redQ1\_5\_1**, indicating whether the number entered is a perfect square or not.

Examples of input and output:

A screenshot of a dialog box titled 'Perfect Square'. It has a close button in the top right. The main area contains the text 'Enter number' above an input field containing the number '9'. At the bottom are 'OK' and 'Cancel' buttons.

A screenshot of an output area. At the top is a button labeled '1.5.1 - Perfect Square'. Below it is a text box containing the message '9 is a perfect square.'

A screenshot of a dialog box titled 'Perfect Square'. It has a close button in the top right. The main area contains the text 'Enter number' above an input field containing the number '15'. At the bottom are 'OK' and 'Cancel' buttons.

A screenshot of an output area. At the top is a button labeled '1.5.1 - Perfect Square'. Below it is a text box containing the message '15 is not a perfect square.'

(6)



**1.5.2 Button [1.5.2 – Sequence of numbers]**

Use the information given below and write code to display the following sequence of numbers:

1 3 9 27 81 243 729
---------------------

- The first term in the sequence is always 1.
- Each subsequent term in the sequence is equal to the previous term multiplied by 3.
- A conditional loop must be used and the loop should stop executing as soon as the sum of the terms in the sequence exceeds the value of 1 000.

(7)

- |                                                                                                                                                                                                                               |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"><li>• Ensure that your examination number has been entered as a comment in the first line of the program file.</li><li>• Save your program.</li><li>• Print the code if required.</li></ul> |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**TOTAL SECTION A: 55**

**SECTION B****QUESTION 2: OBJECT-ORIENTATED PROGRAMMING**

Easy-Secure is a digital certificate-issuing authority (CA). Online traders can apply for a digital certificate. You need a program to test the validity of existing digital certificates and update information on these digital certificates on request.

Do the following:

- Open the incomplete program in the **Question 2** folder.
- Open the incomplete object class **DCertificate\_U.pas**.
- Enter your examination number as a comment in the first line of both the **Question2\_U.pas** file and the **DCertificate\_U.pas** file.
- Compile and execute the program. Currently the program has no functionality.

The following user interface is displayed:

Digital certificate issued by Easy-Secure

Certificate holder  17/10/2017

2.2.1 - Search certificate holder

- Complete the code for this program, as specified in QUESTION 2.1 and QUESTION 2.2.

- 2.1 The incomplete object class (**TDigCertificate**) contains the declaration of four attributes that describe the **DCertificate** object.

The attributes for the **DCertificate** object have been declared as follows:

NAMES OF ATTRIBUTES	DESCRIPTION
fCertHolder	Name of the certificate holder
fExpiryDate	Date the certificate expires
fSecurityCode	Security code compiled using randomly selected hexadecimal values
fIssueNr	The current issue number of the digital certificate

The following incomplete methods have been provided:

**resetExpiryDate, hasExpired and toString**

**NOTE:** The system date is provided and saved in the string variable **sSysDate** in the format 'dd/mm/yyyy'.

- 2.1 Complete the code in the object class, as described in QUESTION 2.1.1 to QUESTION 2.1.6 below.
- 2.1.1 Write code for a constructor method that will receive the name of the certificate holder, the expiry date, the security code and the issue number as parameters. Assign the received parameter values to the respective attributes. (4)
- 2.1.2 Write code for a method called **increaseIssueNr** that will increase the current issue number by 1. (2)
- 2.1.3 Write code to complete the method called **resetExpiryDate** that will use the system date to reset the expiry date attribute of the digital certificate object. The expiry date must be set to one year from the current date (today's date) as it is valid for ONE year only. (4)
- 2.1.4 Write code to complete the method called **hasExpired** that will use the system date to check whether the expiry date has been reached or not and return a Boolean value. (5)
- 2.1.5 Write code for a method called **generateSecurityCode** to compile a new security code to be assigned to the **fSecurityCode** attribute. The code must consist of 10 randomly selected characters from the range 0–9 and A–F, grouped in pairs (two) and separated by colons.
- NOTE:** The final code will consist of 14 characters (10 randomly selected characters and 4 colons).

Example of security code:

9A:D3:23:C6:FA (10)

- 2.1.6 Write code to complete the given **toString** method that will return a string with attributes in the following format:

Digital certificate information:

Certificate holder: <Certificate holder>

Expiry date: <Expiry date>

Security code: <Security code>

Issue number: <Issue number>

(3)

- 2.2 An incomplete unit **Question2\_U** has been provided. It contains code for the object class to be accessible and have an object variable **objDigCert** already declared.

A text file called **DigitalCertificates.txt** contains a list of certificates that have been issued. The details of each certificate holder appears in the following format:

```
<certificate holder>;<issue number>#<expiry date>#
<security code>
```

Example of the details of the first four certificate holders in the text file:

```
GG Technologies;5#01/04/2016#9A:D3:23:C6:FA
JP Scrap Yard;1#01/12/2016#C5:2D:0E:66:A2
Bright Books;11#01/11/2017#18:7F:4B:CD:AA
Creative Jobs;5#01/08/2018#E0:53:CB:C4:22
```

The purpose of the program is to:

- Search for a certificate holder's information in the text file
- Check whether the digital certificate has expired
- Allow the digital certificate to be renewed if the expiry date has been reached

Follow the instructions below to code the solution.

### 2.2.1 Button [2.2.1 – Search certificate holder]

The user must enter the name of the certificate holder. The program must locate the name of the certificate holder in the text file called **DigitalCertificates.txt**.

Write code to do the following:

- Check whether or not the **DigitalCertificates.txt** text file exists. If the text file does NOT exist, display a suitable message and close the program.

- If the text file exists, search for the name of the certificate holder in the text file.
  - If the certificate holder's name is located in the text file:
    - Instantiate the **objDigCert** object.
    - Set the **pnlQ2\_Buttons** to be visible.
  - If the certificate holder's name is NOT located in the text file:
    - Display the message: 'No digital certificate issued previously'.
    - Set the **pnlQ2\_Buttons** to be invisible.

(19)

### 2.2.2 Button [2.2.2 – Display]

Information on the digital certificate object must be displayed in the **redOutput** area using the **toString** method. The output area must be cleared before displaying the information.

Example of output if Bright Books was entered as a certificate holder:

Digital certificate information:	
Certificate holder:	Bright Books
Expiry date:	01/11/2017
Security code:	18:7F:4B:CD:AA
Issue number:	11

(3)

### 2.2.3 Button [2.2.3 – Test validity]

The digital certificate is not valid if the expiry date has been reached. Use the **hasExpired** method written in QUESTION 2.1.4 to determine whether the digital certificate has expired or not.

- Display a suitable message if the digital certificate has NOT expired.
- If the digital certificate has expired, use an InputBox to ask the user whether the certificate must be renewed or not. If the certificate must be renewed, use the **objDigCert** object's methods to:
  - Increase the issue number attribute by 1
  - Generate a new security code
  - Reset the expiry date to a new expiry date

Display information of the digital certificate object in the **redOutput** component using the **toString** method.

Example of output if the digital certificate of GG Technologies has expired and they opted NOT to renew their certificate:

Digital certificate information:	
Certificate holder:	GG Technologies
Expiry date:	01/04/2016
Security code:	9A:D3:23:C6:FA
Issue number:	5

Example of output if GG Technologies opted to renew their digital certificate:

Digital certificate information:	
Certificate holder:	GG Technologies
Expiry date:	01/04/2016
Security code:	38:22:3A:ED:A5
Issue number:	6

**NOTE:** The security code is randomly generated and therefore the values displayed by your program will differ from the values in the screenshots above.

(8)

- Ensure that your examination number has been entered as a comment in the first line of the object class and the form class.
- Save your program.
- Print the code of both the object class and the form, if required.

**TOTAL SECTION B: 58**

**SECTION C****QUESTION 3: PROBLEM-SOLVING PROGRAMMING****SCENARIO**

The online-shopping website of a company called MajorMax allows customers to buy items online from various departments at their store. The manager of the company must analyse their weekly sales figures.

Do the following:

- Open the incomplete program in the **Question 3** folder.
- Enter your examination number as a comment in the first line of the **Question3\_U.pas** file.
- Compile and execute the program. Currently the program has no functionality.

**Supplied GUI:**

The GUI below represents the interface of the program used by MajorMax to keep track of their weekly sales figures.



- Complete the code for each question, as described in QUESTION 3.1 to QUESTION 3.3.

**NOTE:**

- Good programming techniques and modular design must be applied in the design and coding of your solution.
- You may NOT change the code provided.

The program contains code that declares two arrays, **arrDepartments** and **arrSales**.

The **arrDepartments** array contains the names of the various departments that sell products online.

Code declaring the **arrDepartments** array:

```
arrDepartments: array[1..8] of String = (
    'PCs & Notebooks', 'Tablets & eReaders',
    'Software', 'Printers, Toners and Ink', 'Cellphones',
    'Games & Drones', 'Network Equipment', 'Accessories');
```

The **arrSales** array is a two-dimensional array that contains the sales figures for the first six weeks of the year for each department. The rows in the array represent the various departments and the columns represent various weeks.

Code declaring the **arrSales** array:

```
arrSales: array[1..8, 1..6] of Real = (
    (935.89, 965.99, 4056.77, 5023.89, 3802.66, 1146.98) ,
    (2667.78, 2491.78, 1989.65, 2647.88, 1601.56, 1921.99) ,
    (6702.45, 4271.56, 3424.45, 3924.55, 3085.45, 3359.77) ,
    (6662.34, 6658.45, 8075.43, 2360.66, 2635.44, 7365.69) ,
    (16405.33, 9741.37, 13381.56, 18969.76, 8604.55, 20207.56) ,
    (10515.29, 7582.66, 9856.56, 7537.68, 9115.67, 8401.55) ,
    (7590.99, 9212.65, 9070.98, 6439.99, 7984.88, 8767.45) ,
    (9220.65, 8097.99, 10067.44, 9960.87, 10109.56, 6571.66) );
```

### 3.1 Button [3.1 – Sales information]

Display the content of the **arrSales** array with suitable headings in the output component provided. All monetary values must be displayed in currency format with TWO decimal places.

Example of output:

Department	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6
PCs & Laptops	R 935.89	R 965.99	R 4 056.77	R 5 023.89	R 3 802.66	R 1 146.98
Tablets & eReaders	R 2 667.78	R 2 491.78	R 1 989.65	R 2 647.88	R 1 601.56	R 1 921.99
Software	R 6 702.45	R 4 271.56	R 3 424.45	R 3 924.55	R 3 085.45	R 3 359.77
Printers, Toners and Ink	R 6 662.34	R 6 658.45	R 8 075.43	R 2 360.66	R 2 635.44	R 7 365.69
Cellphones	R 16 405.33	R 9 741.37	R 13 381.56	R 18 969.76	R 8 604.55	R 20 207.56
Games & Drones	R 10 515.29	R 7 582.66	R 9 856.56	R 7 537.68	R 9 115.67	R 8 401.55
Network Equipment	R 7 590.99	R 9 212.65	R 9 070.98	R 6 439.99	R 7 984.88	R 8 767.45
Accessories	R 9 220.65	R 8 097.99	R 10 067.44	R 9 960.87	R 10 109.56	R 6 571.66

(7)



### 3.2 Button [3.2 – Display underperforming departments]

A report of all underperforming departments per week is required. A department is underperforming when their sales figure is lower than the average sales for all the departments for that week.

Display the report in the output component provided, with suitable headings. All monetary values must be displayed in currency format with TWO decimal places.

Example of output for the first three weeks using the original data:

Underperforming departments per week:	
Week 1: Average sales figure: R 7 587.59	
PCs & Laptops	R 935.89
Tablets & eReaders	R 2 667.78
Software	R 6 702.45
Printers, Toners and Ink	R 6 662.34
Week 2: Average sales figure: R 6 127.81	
PCs & Laptops	R 965.99
Tablets & eReaders	R 2 491.78
Software	R 4 271.56
Week 3: Average sales figure: R 7 490.35	
PCs & Laptops	R 4 056.77
Tablets & eReaders	R 1 989.65
Software	R 3 424.45

(14)

### 3.3 Button [3.3 – New week]

Currently the data in the **arrSales** array represents the sales figures for the first six weeks of the year. When sales figures for a new week, for example Week 7, need to be analysed and therefore recorded in the array, the current data for Week 1 in the array must be backed up in a text file. The name of the text file is the number of the week of the sales figures that are archived. Example: If the sales figures for Week 1 are archived in the file, then the name of the text file will be **'Week 1.txt'**.

When the data for Week 1 has been archived, the data for Week 2 in the **arrSales** array must be moved to the position of Week 1 in the array; the data for Week 3 must be moved to Week 2, and so on.

For test purposes the sales data for the new week must be randomly generated in the range R500–R5 000.

Example of output if data for a new week has been added to the array and the data of for the first week has been archived:

Department	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
PCs & Laptops	R 965.99	R 4 056.77	R 5 023.89	R 3 802.66	R 1 146.98	R 3 077.47
Tablets & eReaders	R 2 491.78	R 1 989.65	R 2 647.88	R 1 601.56	R 1 921.99	R 4 862.12
Software	R 4 271.56	R 3 424.45	R 3 924.55	R 3 085.45	R 3 359.77	R 3 850.06
Printers, Toners and Ink	R 6 658.45	R 8 075.43	R 2 360.66	R 2 635.44	R 7 365.69	R 2 585.94
Cellphones	R 9 741.37	R 13 381.56	R 18 969.76	R 8 604.55	R 20 207.56	R 2 345.37
Games & Drones	R 7 582.66	R 9 856.56	R 7 537.68	R 9 115.67	R 8 401.55	R 3 296.11
Network Equipment	R 9 212.65	R 9 070.98	R 6 439.99	R 7 984.88	R 8 767.45	R 1 218.38
Accessories	R 8 097.99	R 10 067.44	R 9 960.87	R 10 109.56	R 6 571.66	R 1 206.24

**NOTE:** The labels used to display the weeks will be updated with code to reflect the number of the new week that was added. The data that your program displays for the new week may differ from the values in the screenshot due to random values used.

Example of the contents of the **'Week 1.txt'** text file:

```
PCs & Laptops: R 935.89
Tablets & eReaders: R 2 667.78
Software: R 6 702.45
Printers, Toners and Ink: R 6 662.34
Cellphones: R 16 405.30
Games & Drones: R 10 515.30
Network Equipment: R 7 590.99
Accessories: R 9 220.65
```

Example of output if data for the next new week has been added to the array and the data for the second week has been archived:

Department	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8
PCs & Laptops	R 4 056.77	R 5 023.89	R 3 802.66	R 1 146.98	R 3 077.47	R 4 540.88
Tablets & eReaders	R 1 989.65	R 2 647.88	R 1 601.56	R 1 921.99	R 4 862.12	R 1 343.29
Software	R 3 424.45	R 3 924.55	R 3 085.45	R 3 359.77	R 3 850.06	R 922.19
Printers, Toners and Ink	R 8 075.43	R 2 360.66	R 2 635.44	R 7 365.69	R 2 585.94	R 4 412.70
Cellphones	R 13 381.56	R 18 969.76	R 8 604.55	R 20 207.56	R 2 345.37	R 725.92
Games & Drones	R 9 856.56	R 7 537.68	R 9 115.67	R 8 401.55	R 3 296.11	R 2 521.59
Network Equipment	R 9 070.98	R 6 439.99	R 7 984.88	R 8 767.45	R 1 218.38	R 2 943.60
Accessories	R 10 067.44	R 9 960.87	R 10 109.56	R 6 571.66	R 1 206.24	R 3 711.95

Example of the contents of the **'Week 2.txt'** text file:

```
PCs & Laptops: R 965.99
Tablets & eReaders: R 2 491.78
Software: R 4 271.56
Printers, Toners and Ink: R 6 658.45
Cellphones: R 9 741.37
Games & Drones: R 7 582.66
Network Equipment: R 9 212.65
Accessories: R 8 097.99
```

(16)

- Ensure that your examination number has been entered as a comment in the first line of the program file.
- Save your program.
- Print the code if required.

**TOTAL SECTION C: 37**  
**GRAND TOTAL: 150**