GAUTENG PROVINCE
EDUCATION
REPUBLIC OF SOUTH AFRICA

PREPARATORY EXAM 2015

COMMON EXAM

GRADE 12

QUESTION PAPER

SUBJECT	**:**	**IT PAPER 1**
TIME	**:**	**3 HOURS**
MARKS	**:**	**150**

This question paper consists of 26 pages

INSTRUCTIONS AND INFORMATION

1. This question paper is divided into THREE sections. Candidates must answer 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 paper is set in Delphi.

4. Make sure that you answer the questions according to the specifications that are given in each question. Marks will only 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 the programming language for any of these routines.

8. All data structures must be defined by you, the programmer. You may not use components provided within the interface to store and later retrieve data.

9. You must save your work regularly on the disk you have been given, or the disk space allocated to you for this examination.

10. Make sure that your name and surname appears as a comment in every program that you code.

11. When required, print the programming code of all the programs/classes that you completed/attempted. 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. Ensure that all files can be read.

Copyright reserved

13. The following files are included in your data folder named **Prelim Data (ENG)**:

Question1:

Question1_P.dpr
Question1_P.res
Question1_U.dfm
Question1_U.pas
www.wheelerpackaging.co.uk.jpg

Question2:

Claim_U.pas
Question2_P.dpr
Question2_P.res
Question2_U.dfm
Question2_U.pas
ValueCars.txt

Question3:

my_cover.jpg
new_health.jpg
u_save.jpg
Question3_P.dpr
Question3_P.res
Question3_U.dfm
Question3_U.pas

SECTION A

QUESTION 1: GENERAL PROGRAMMING SKILLS

SCENARIO

E-tailing businesses (electronic retailers) depend on customer details and packaging. You have been assigned to assist in the programming requirements of such a business entity.

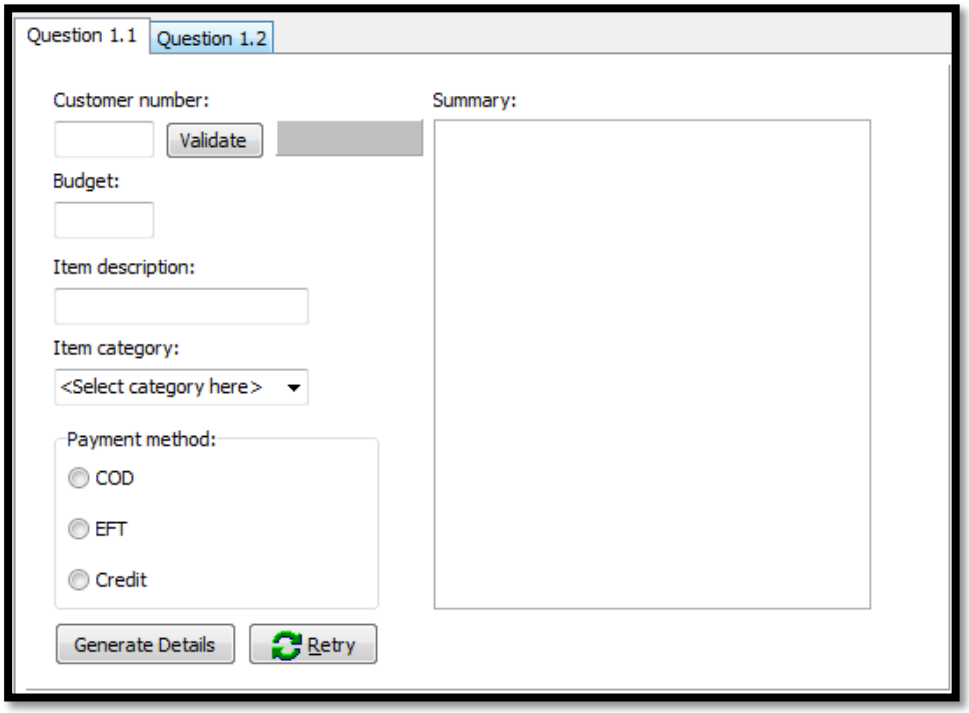
INSTRUCTIONS:

- The project **Question 1** is provided in the **Prelim Data (ENG)** folder provided.
- Open the incomplete project file **Question1_P.dpr** in the **Question 1** folder.
- Add your NAME and SURNAME as a comment in the first line of the main form unit **Question1_U.pas**.

Do the following:

- Compile and execute the program.
- The interface displays TWO tab sheets namely:
  - Question 1.1 (*e-tailer customer details summary*)
  - Question 1.2 (*parcel cost calculation*)
- Complete the code for each tab as described in QUESTION 1.1 and QUESTION 1.2.

1.1 Interface for the Question1.1 tab:



1.1.1 **[Button – Generate Details]**

Use the information from the THREE edits, ONE combo box and ONE radio group to compile the following report. The report must be displayed in the Rich Edit (**redDisplay**). Write code to do the following:

- The rich edit must be cleared once the event is initiated.
- The report must start with a suitable heading indicating the heading *Client Summary* at the top of the rich edit component followed by an open line in order to separate the heading from the rest of the details.
- The client summary must start with the details of the client number which follows a suitable description (*Customer number*). The customer number must be stored into a variable with **class/unit scope**.
- The client budget must be displayed after the customer number in the following line with a suitable heading (*Client budget*). The currency must be displayed as *R* (Rand) with two decimals.
- The item description must be displayed on the following line. It combines the item category followed by a ' – ' and the item description entered by the user. Use the heading *Purchase item*.
- The payment method will be listed which is preceded by a suitable description heading called *Payment method*.

- If the payment method is COD (cash on delivery), then a suitable message must be displayed in the rich edit stating that the client must contact the courier personally for payment arrangements.

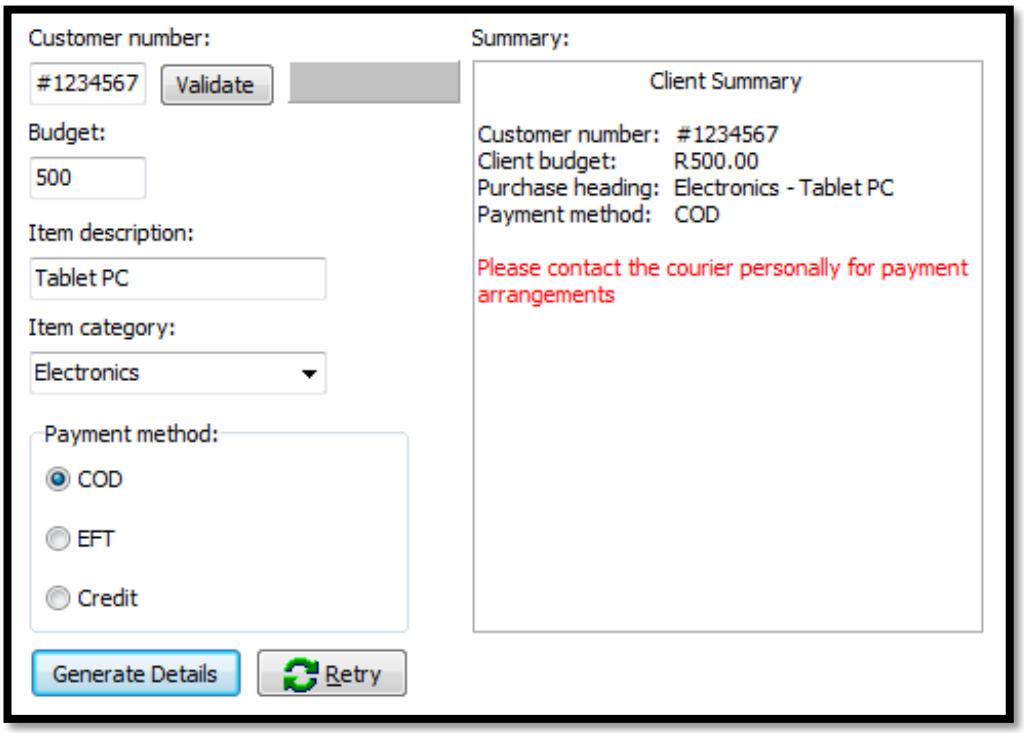
- If the method of payment is credit card, then a message should be displayed stating that the user must be warned of an interest of 15% charged for goods sold on credit.

- If the method is via EFT, then a message must be displayed informing the user to obtain the correct banking details on the business website.

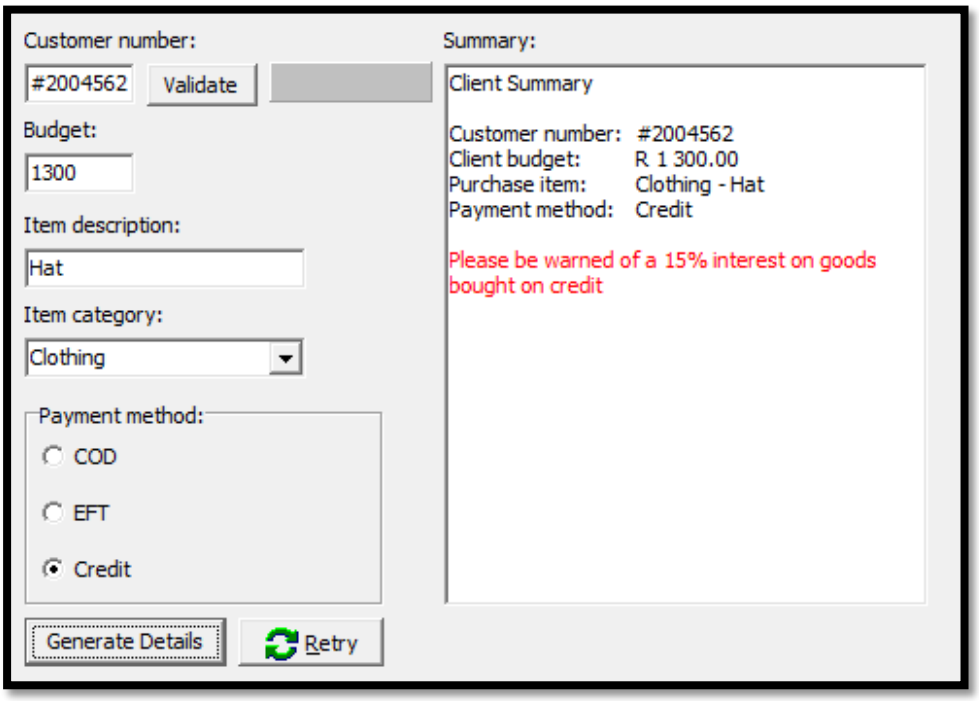
NOTE: Please ensure that all information is aligned properly.

Examples of compiled client summaries:

Example of output if a customer with customer number #1234567, having a budget of R500 wants to buy a Tablet PC from the Electronics category and payment is COD:



Example of output if a customer with customer number #2004562, having a budget of R1300 wants to buy a hat from the Clothing category and payment is Credit:



(17)

1.1.2 **[Button – Validate]**

Write code that will validate the customer number according to the following criteria:

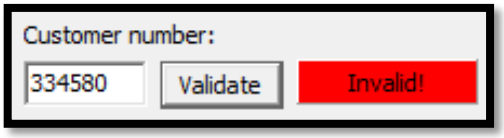
CRITERIA	DESCRIPTION	MESSAGE IF NOT MET
Criteria 1: Customer number length	8 characters only	Length!
Criteria 2: First character	#	Start: #

If the customer number is valid according to both criteria above, the panel component (**pnlValidate**) will change to the colour lime with a suitable caption (e.g. *Valid!*)

If none of the criteria is satisfied the panel must appear red and the message that the customer number is *Invalid* must be displayed.

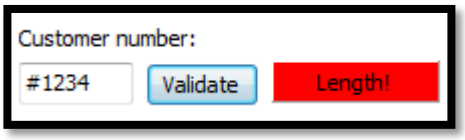
If either one of the criteria is not met, a suitable message must be displayed also in colour red.

Example of an invalid customer number:



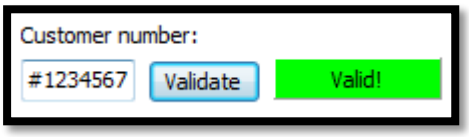
Example of an invalid customer number due to the first character not being a #:

Example of an invalid customer number due to the length of the number:



(8)

Example of a valid customer number:

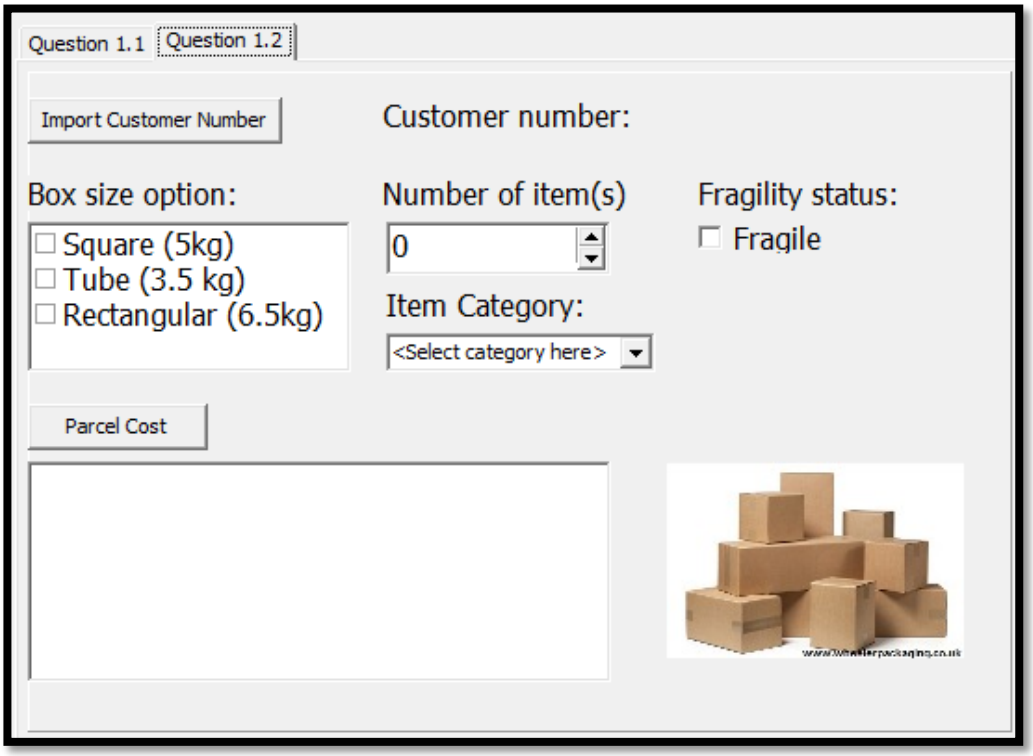


1.1.3 **[Button – Retry]**

Write code for the retry button that will do the following:

- Clear the edit components.
- Set the text of the combo box to *<Select category here>*
- Ensure that the radio group component does not have any options selected.
- Set the validation panel component to colour silver with no caption.
- The component displaying the summary of details must be cleared of all information.
- The cursor must be set to the first edit (awaiting a new customer number). (4)

1.2 Interface for the Question 1.2 tab:



1.2.1 **[Button – Import Customer Number]**

Write code that will display the customer number that was obtained and validated in **question 1.1.2**. Display the customer number in the label (*lblCustNum*) provided. (2)

1.2.2 **[Buttons – Parcel Cost]**

The button will obtain the following information from the components provided:

- **Box size**
  - Square boxes can carry 5 kg each costing R55.95 per box.
  - Tube shaped boxes can carry 3.5 kg each costing R39.35 per box.
  - Rectangular boxes can carry 6.5 kg each costing R71.45 per box.

- **Amount of items**
  - Contains the number of items that will be packaged.

- **Item category**
  - The type of items that will be packaged. Each type of item consists of a weight factor which is tabulated below:

Item category:	Weight factor:
Clothing	1.3 kg per item
Motoring	10.7 kg per item
Electronics	2.5 kg per item
Gardening	5.8 kg per item
Groceries	0.3 kg per item

- **Fragility status**
  - Indicates whether the package should be handled with care or not. If the parcel is fragile, then the final cost is increased by 5%.

The total parcel cost is calculated as follows:

- Step 1: Determine the total weight by multiplying the **number of items** by the **weight factor**.

- Step 2: Determine the number of boxes required according to the size of the box selected.

  NOTE: if the total weight is 3.9 kg for example and the user selected a Tube box size (3.5 kg), then TWO of such boxes must be used.

  HINT: make use of a built-in custom subroutine that will always round off to the highest integer or make use of integer division techniques in order to achieve the same outcome.

- Step 3: Calculate the final cost of the parcel according to the price per box. The cost must be adjusted accordingly should the user mark the parcel as fragile!

- Step 4: Display the cost to send the parcel with a suitable message in the output area.

The **Parcel Cost** button will store the calculated total cost for the parcel into the variable **TotalCost** that is given with class scope in the main unit.

Instructions:

- Create ONE custom subroutine called **DetermineCost** that will do the following, in the given order:

  - Determine the cost of the parcel:

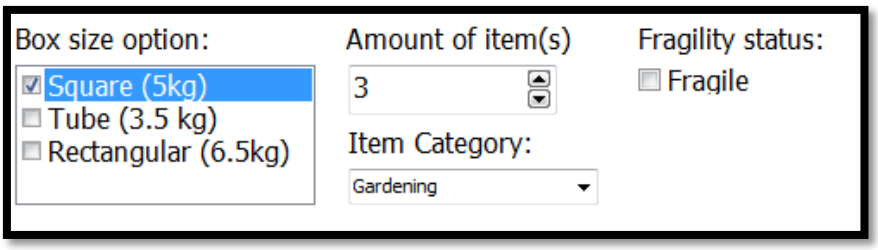
    Make use of value parameters for the box size, category and fragility in order to determine the total cost which must be returned as reference parameter **TotalCost** (already declared with **class scope** in the main unit)

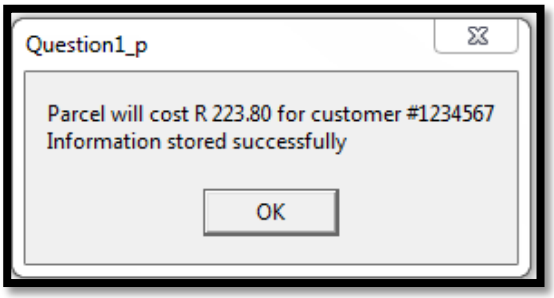
  - Refresh all components:

    All components used for the parcel calculation must be refreshed (cleared) in order and priority (focus) must be set on the box sizes component.

- Write code that will call **DetermineCost** successfully and display a suitable message for the parcel cost with the customer's number included.

Example of output when a user wants to send a parcel containing 3 items of gardening using a square box:

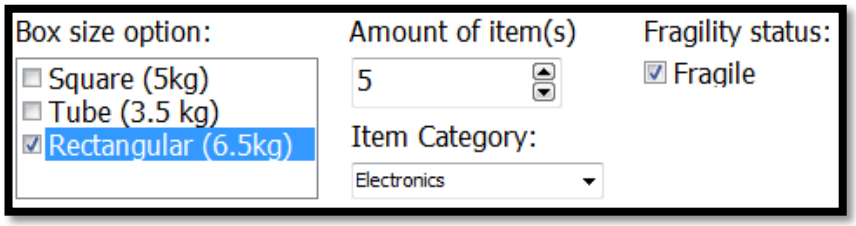


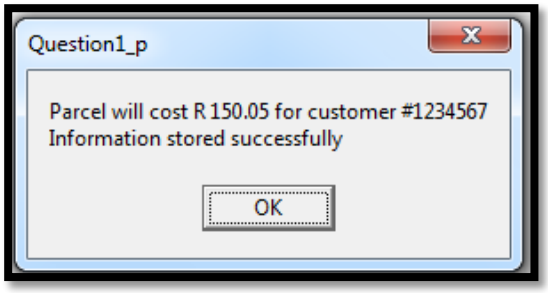


NOTE: Example of the calculations for the parcel in example one:

3 items x weight factor (**5.8kg** per item) = **17.4kg**
Box Size chosen: Square (5kg) @ R55.95 per box
Thus, a **minimum** of FOUR whole square boxes needed (**20kg**)
Final Cost = R55.95 x 4 = **R223.80** (note: not fragile)

Example of output when fragile items are to be sent:





NOTE: Example of the calculations for the parcel in example two: (18)

5 items x weight factor (**2.5kg** per item) = **12.5kg**
Box Size chosen: Rectangular (6.5kg) @ R71.45 per box
Thus, a **minimum** of TWO whole rectangular boxes needed (**13kg**)
Final Cost = R71.45 x 2 = R142.90 x 5% **= R150.05** (fragile parcel, thus a 5% increase)

TOTAL SECTION A: 49

SECTION B

QUESTION 2: OBJECT-ORIENTATED PROGRAMMING

SCENARIO

Car insurance is an unavoidable necessity to many motorists who try to avoid personal liabilities involving any vehicle accidents related to unforeseen circumstances.

Once an insured motorist is involved in an accident, the following must be considered:

- The value of the vehicle on the date of the accident.
- The assessed (initial) damage done to the vehicle.
- The liability fee that the motorist (insurance holder) is responsible for, based on:
  - his/her age
  - state of soberness
  - cost of damage to the vehicle
  - the severity of the accident

Monthly insurance premiums are also determined according to standard criteria for insurance holders.

INSTRUCTIONS:

The project **Question2_P** is provided in the **Prelim Data (ENG)** folder which also contains:

- A main file called **Question2_U.pas**
- An incomplete unit file called **Claim_U.pas**
- A text file containing depreciation values called **ValueCars.txt**

Open the incomplete project file **Question2_P.dpr** in the **Question 2** folder.

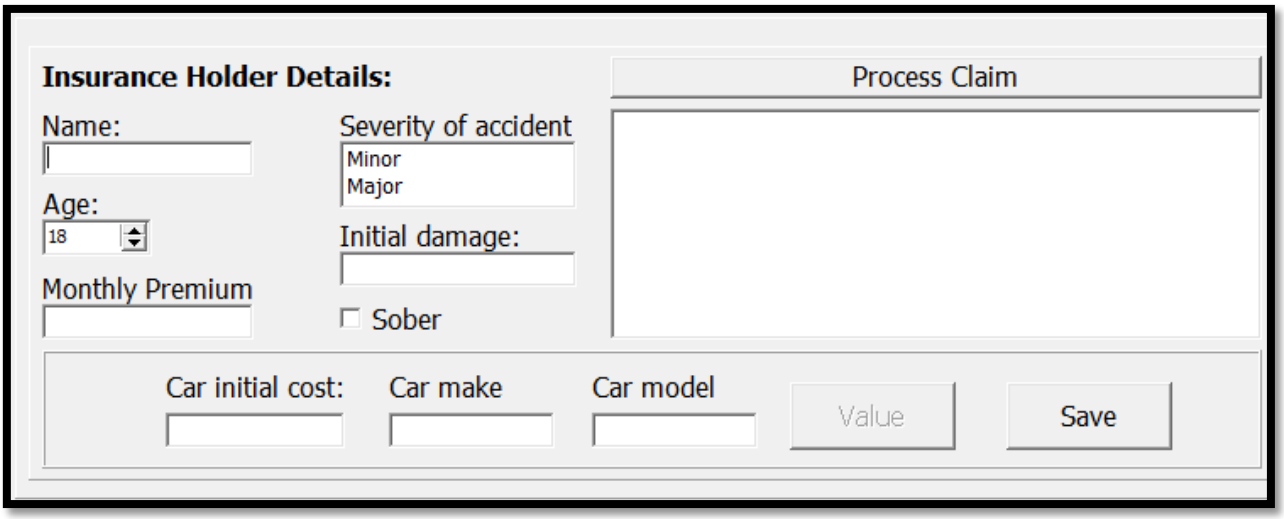
Add your NAME and SURNAME as a comment in the first line of the main form unit **Question2_U.pas** as well as the separate unit **Claim_U.pas**.

Do the following:

- Compile and execute the program. Note that neither the main form unit nor the class has any functionality.

- Complete the instructions as described in QUESTION 2.1 and QUESTION 2.2 in order to add functionality to the program.

- The GUI supplied by the main form looks as follows:



2.1 Complete the code in the insurance claim class **TClaim** as described below in the following questions.

2.1.1 Write code to add the following five attributes to the class (remember to choose suitable data types according to the description of each attribute):

Description	Attribute
Age of insurance holder	fAge
Severity of the accident (minor/major)	fSeverity
Was the insurance holder sober or not?	fSober
Monthly insurance premium	fPremium
Initial damage done to the vehicle	fInitialDamage

(3)

2.1.2 Write code in order to create a constructor which will contain the following parameters:

- Age
- Severity
- Soberness
- Premium

You are required to initialise all relevant attributes using the parameter values provided. Remember that given attributes must be properly initialised as part of the constructor definition in the class. (3)

2.1.3 The declarations of four accessor-methods have been provided. Remove the comment symbols preceding each one and write code to implement each method correctly. (4)

2.1.4 Write a mutator-method called **SetDamage** for the **fInitialDamage** attribute. (2)

2.1.5 The following method has been provided: **CalcClaim**

Complete the code for this method to calculate the amount that the motorist (insurance holder) can claim from the insurance company.

The method must first determine the liability fee. This fee is the motorist's contribution towards repairs of the vehicle. The liability fee is based on the following criteria:

- If the motorist is 25 years or older and the severity of the accident is minor the liability fee will be 50% of the monthly insurance premium otherwise the liability fee will be twice the monthly premium.

- If the insurance holder was not sober, the liability fee will increase by an additional 25%.

Once the liability fee is calculated, the claim can be determined. The claim is calculated as the difference between the initial damage and the liability fee. The motorist will pay the total initial damage should the liability fee be equal to or greater than the initial damage, thus the claim will be zero. (8)

2.1.6 Write a method called **CalcCurrentVal** to calculate and return the current value of a vehicle. The method must receive the vehicle's initial cost, annual depreciation and model as parameters.

NOTE that the model of a vehicle indicates the year in which it was manufactured.

The current value of a car is determined as follows:

Current value = Initial cost -
[(current year - model) x annual depreciation]

NOTE that the current value cannot be negative. The lowest possible value is zero. (4)

2.1.7 Complete the existing **toString** method that will return the following formatted summary:

```
Age: <motorist's age>
Severity: <severity of accident>
Sober: <state of soberness Y/N>
Premium: <monthly insurance premium>
Initial damage: <initial damage done to the vehicle>
```

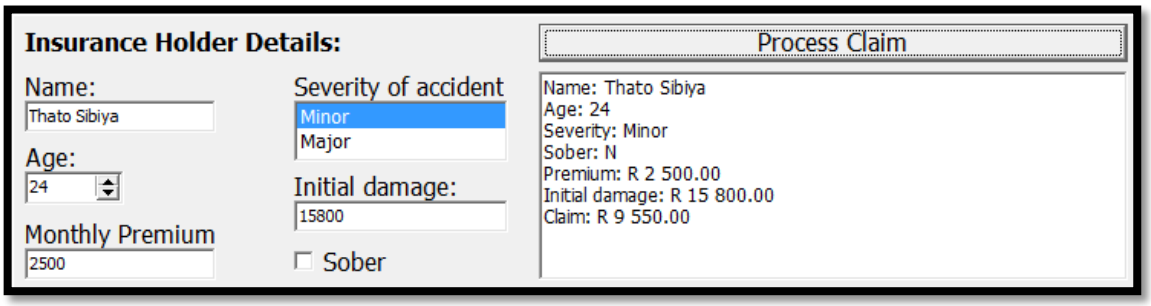
(3)

2.2 Do the following to complete the code for the buttons in the main form unit:

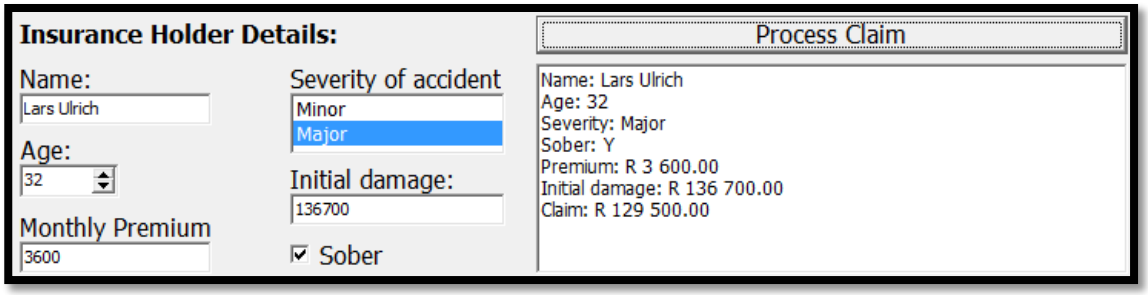
2.2.1 **[Button – Process Claim]**

- Write code to display (**rich edit**) the insurance holder's name from data in the component provided.
- Use the components providing data for the age, monthly premium, accident severity, sober state and initial damage to instantiate a new **TClaim** object. Use the **Claim** object variable that has been declared globally as part of the given code to store the object.
- Display the details of the insurance holder using the **toString** method.
- Display the total amount that will be claimed.
- Enable the **Value** button.

Example of an insurance holder called Thato Sibiya (age 24) with a monthly insurance premium of R2500.00 who was involved in a minor accident with an initial assessed damage of R15 800.00. **NOTE**: The insurance holder was not sober.



Example of an insurance holder called Lars Ulrich (age 32) with a monthly insurance premium of R3600.00 who was involved in a major accident with an initial assessed damage of R136 700.00. **NOTE**: The insurance holder was sober.



(9)

2.2.2 **[Button - Value]**

You have been provided with a text file named **ValueCars.txt**.

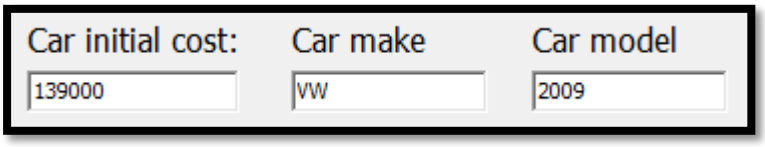
The text file structure is as follows:

```
<car make>#<annual depreciation value of the car>
```

Example of the contents of the text file:

```
VW#3500
TOYOTA#3400
NISSAN#3300
HONDA#3350
DIAHATSU#3550
ISUZU#3100
FORD#3250
BMW#3050
MERCEDES#3175
```

The user must enter the initial cost, make and model of the car using the components provided as illustrated in the following example:



Write code that will do the following:

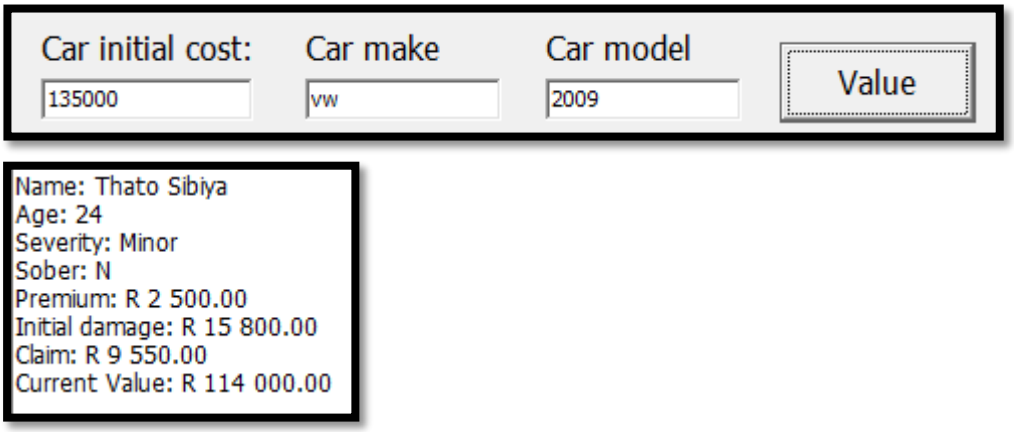
- Check whether the text file **ValueCars.txt** exists. Display a suitable message if the text file does not exist and leave the event **btnValueClick**.

- Use a conditional loop and search for the car make only that was obtained from the second edit component in the previous example. Ensure that your search is not case sensitive.

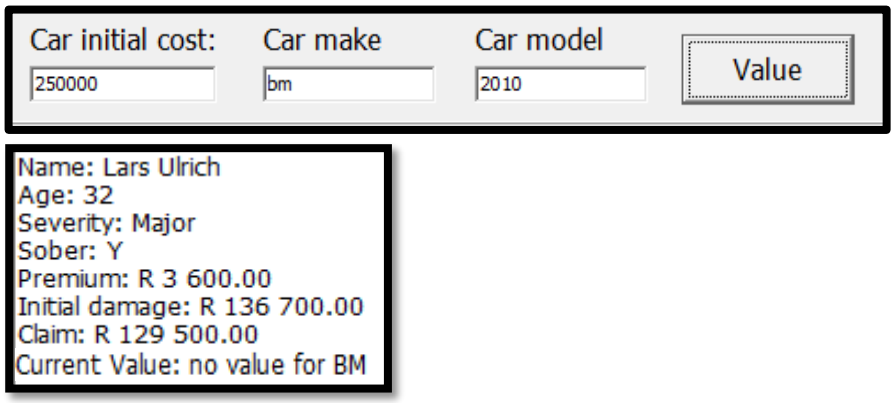
- If the car make (e.g. VW) is found:
  - Use the data from the line to extract the correct annual depreciation.
  - Determine the current value of the car by using the initial car cost, annual depreciation and car model as parameters for the **CalcCurrentVal** method of the **Claim** object created in question 2.2.1.
  - Display the current value of the car in the rich edit provided.

- If the car make is NOT found, add a suitable message in the rich edit component provided (e.g. Current value: no value <car make not found>).

Example of output if car make is found in text file:



Example of output if car make is not found:



(15)

2.2.3 **[Button - Save]**

Write code that will store important details of the claim information to a text file. Do the following:

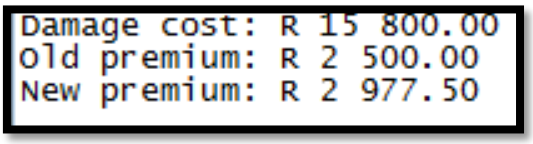
- Create a text file using the name of the person who processed the claim followed by '-' and the amount that is claimed as the file name (e.g. *Lars Ulrich – R 129 500.00.txt*)

- The text file must store the damage cost as well as the old and new monthly premium.
  - The old monthly premium is obtained from the information that the user provided.
  - The new monthly premium will be calculated as the old premium plus 5% of the claimed amount.

- Display a suitable message that will indicate the text file has been saved and also indicate the new monthly premium.

Example of a correct message output:

Example of correctly saved text file:



(7)

TOTAL SECTION B: 58

SECTION C

QUESTION 3: PROBLEM-SOLVING

SCENARIO

A small business with 10 employees was approached by three medical aid companies that are searching for new members. Medical aid is a form of health insurance used in the event of injury or illness. These companies make use of an interesting scale system in order to determine a member's monthly medical insurance premium.

INSTRUCTIONS:

The project **Question 3** is provided in the **Prelim Data (ENG)** folder provided.
Open the incomplete project file **Question3_P.dpr** in the **Question 3** folder.
Add your NAME and SURNAME as a comment in the first line of the main form unit **Question3_U.pas**.

Do the following:

- Compile and execute the program.
- Complete the code as described in QUESTION 3.1, QUESTION 3.2 and QUESTION 3.3 in order to add functionality to the program.

The following interface has been provided:

3.1 **[Form OnActive]**

The interface is incomplete. Images for the first two medical aids appear on the GUI except for New Health medical aid. Write code that will dynamically create an image for New Health in the OnActive event of the form.

NOTE: The image *new_health.jpg* has been provided under the QUESTION 3 folder.

Parent	:	frmQ3
Width	:	193
Height	:	113
Top	:	8
Left	:	400
LoadFromFile	:	new_health.jpg

(5)

3.2 The employees (clients) will use the program to look at the monthly premium that every medical aid has to offer. The main unit contains TWO existing arrays called **arrCompany** and **arrClients**.

The descriptions of these arrays are as follows:

arrCompany	Description: This array contains the scale fractions for all factors that affect a member's monthly premium. These factors include the member age, gender and number of children. Each factor is used in the calculation of the monthly premium. Structure: **<age factor>M<male factor>F<female factor>C<child factor>** Example of an array element: **0.21M5.1F4.3C1.5** Age factor = 0.21 Male points = 5.1 Female points = 4.3 Child factor = 1.5 **NOTE**: **arrCompany** consists of THREE entries for medical aids U-Save, My Cover and New Health in order from element 1, 2 and 3.
arrClients	Description: This array contains the following client information: surname; age; gender; number of children and monthly salary. Structure: **<surname>$<age>$<gender><number of children>$<salary>** Example of an array element: **Anderson36M3$8750** Surname = Anderson Age = 36 Gender = M Number of children = 3 Monthly salary = R8,750.00

3.2.1 Add a suitable output component on the **pnlOutput** component provided. This component must either be a string grid or a memo. (1)

3.2.2 **[Button – List Premiums]**

You must write code that will display the monthly premium as determined by each medical aid according to each member's age, gender, number of children and salary.

Instructions:

- The main unit contains a user-defined function called **Scale** that has already been declared and implemented. The function consists of FOUR parameters namely age, gender, children and number for medical aid.

  Complete the code in the function definition in order to calculate the scale for any client.

  NOTE that the number parameter indicates the medical aid option (1 – U Save, 2 – My Cover and 3 – New Health)

  The calculation of the scale for each medical aid must be calculated by using the following model:

Criteria	**Calculation**	**Add / Subtract from scale**
Age	Age of member * Age factor	Add
Gender	Gender points	Add
Children	Number of children * Child factor	Subtract

Example of a scale calculation for a lady who is 33 years old with two children, applying for a medical aid with an age and child factor of 0.25 and 1.8 as well as gender points (females) of 4.1.

Age	33 * 0.25	+ 8.25
Gender	4.1	+ 4.1
Children	2 * 1.8	- 3.6
Final scale (8.25+4.1-3.6) = **8.75**		

- Write code for the List Premiums button that will do the following:

  - Extract the surname, age, gender, number of children and salary from each **arrClients** entry.

  - Determine the scale by using the **Scale** function.

  - Calculate the monthly premium for each medical aid for each client. This will give THREE premiums per client. The monthly premium is calculated as follows:

The scale must be converted to a percent fraction (scale ÷ 100) after which it must be multiplied by the client's monthly salary.

Example of the same woman (see previous example) having a monthly salary of R9,450.00:

Monthly premium	=	(8.75 ÷ 100) * 9450
	=	0.0875 * 9450
	=	**R826.88** per month

- Store the client name and three calculated medical aid premium options into a new array declared globally. Declare the array as **arrPremiums**.

  NOTE: you may declare **arrPremiums** either as a one or two dimensional array.

  HINT: Consider your output component.

- Display each client's premium for each medical aid option in the string grid component provided. You must provide suitable headings for each medical insurance company as well.

Example of correct output in a string grid component:

	U Save	My Cover	New Health
Anderson	R 714.00	R 957.25	R 504.00
Kekana	R 940.68	R 1 078.92	R 1 464.48
Gouws	R 454.92	R 609.28	R 390.32
Nhleko	R 547.39	R 558.11	R 664.64
Campbell	R 2 067.97	R 2 937.03	R 2 079.35
Thusi	R 1 082.72	R 1 222.08	R 1 712.52
Vermaak	R 2 051.40	R 2 628.60	R 1 024.40
Clapton	R 3 082.38	R 4 238.72	R 3 171.88
Tankian	R 606.24	R 843.36	R 613.44
Mahlangu	R 457.62	R 394.98	R 614.22

NOTE that a memo could also be used to display the same information displayed in the example above. (19)

3.2.3 **[Button - Average]**

Write code that will calculate the average monthly premium using the data stored in **arrPremiums**. Display the averages for each medical insurance company by adding it after the data displayed in question 3.2.2.

Example of correct averages:

	U Save	My Cover	New Health
Anderson	R 714.00	R 957.25	R 504.00
Kekana	R 940.68	R 1 078.92	R 1 464.48
Gouws	R 454.92	R 609.28	R 390.32
Nhleko	R 547.39	R 558.11	R 664.64
Campbell	R 2 067.97	R 2 937.03	R 2 079.35
Thusi	R 1 082.72	R 1 222.08	R 1 712.52
Vermaak	R 2 051.40	R 2 628.60	R 1 024.40
Clapton	R 3 082.38	R 4 238.72	R 3 171.88
Tankian	R 606.24	R 843.36	R 613.44
Mahlangu	R 457.62	R 394.98	R 614.22
AVERAGE	R 1 200.53	R 1 546.83	R 1 223.93

NOTE that a memo could also be used to display the same information displayed in the example above. (5)

3.3 The program must indicate which medical aid will provide the lowest monthly premium for every client. In addition, the medical aid companies would like to know the total number of clients that would most likely become a member due to the lowest premium offer. All the medical aid companies made the suggestion that a month with no premium must be awarded to one lucky client in order to attract the users to the program.

3.3.1 **[Button – Best Medical Aid Option]**

Write code that will calculate and display the following:

- The best medical aid option (company name) for each client. The output must be displayed in the memo component (**memDisplay**) provided and must include the client's name as well as the medical aid company which offers the lowest monthly premium.
- Pick a random client and display that he/she has won a month with no premium.
- Calculate the number of clients per medical aid.

Example of correct output:

```
Anderson        New health
Campbell        U Save
Clapton         U Save
Gouws           New health
Kekana          U Save
Mahlangu        My Cover
Nhleko          U Save
Tankian         U Save
Thusi           U Save
Vermaak         New health
Tankian wins 1 month zero premium!
```

(11)

3.3.2 **[Button – Totals]**

Write code that will display the number of clients per medical aid (calculated in question 3.3.1) in the panel provided.

Example of correct output:

```
U Save: 6 | My Cover: 1 | New Health: 3
```

(2)

TOTAL SECTION C: 43

GRAND TOTAL: 150

Copyright reserved