higher education & training

Department:
Higher Education and Training
REPUBLIC OF SOUTH AFRICA

NATIONAL CERTIFICATE (VOCATIONAL)

PRINCIPLES OF COMPUTER PROGRAMMING

(Second Paper)

NQF LEVEL 3

(10041023)

13 March 2018 (X-Paper)
09:00–13:00

This question paper consists of 17 pages.

NCV

TIME: 4 HOURS
MARKS: 100

INSTRUCTIONS AND INFORMATION

1. Answer ALL the questions.
2. Read ALL the questions carefully.
3. Number the answers according to the numbering system used in this question paper.
4. Write neatly and legibly.

INSTRUCTIONS TO STUDENTS AND LECTURERS

1. Candidates MUST be seated in the computer laboratory 30 minutes before the start of the examination to ensure that all computers have the appropriate programs.
2. These questions must be completed on a computer by using MS Office 2003 or a later version and Visual Studio.NET 2005 or VB.NET 2005 Express or later versions.
3. Candidates must make sure that they print their work immediately after they complete a question. The candidates' EXAMINATION NUMBER must be on ALL printouts. NO questions without a printed EXAMINATION NUMBER will be marked.
4. In the event of problems such as a power failure, computer breakdown or a printer breakdown, the invigilator will make the necessary arrangements to continue with the examination process once the problem has been resolved. Students will not be penalised with time lost under these circumstances.
5. All work on the computer should be saved at regular intervals to prevent loss of work. NO additional time may be allowed for work lost due to lack of saving.

Copyright reserved

6. No candidate may print his/her work for another candidate, make a memory stick available to another candidate or access another candidate's work on the network. Any attempt to access any information from or transfer information to another candidate in whatever manner is a contravention of the examination rules and regulations and will be viewed in an extremely serious light.

7. ALL PRINTOUTS and screen prints done during the session MUST be handed in.

8. Candidates are NOT allowed to copy the source code from VB.NET to MS Word and then print it from the word processor. The source code must be printed directly from VB.NET.

9. Steps to print a VB.NET form or any other screen:

    9.1 Run/Debug the solution.

    9.2 Hold the ALT key down and press the print screen button. (This will copy the active screen.)

    9.3 Open MS Word and paste the form/screen in a new document.

    9.4 Insert your EXAMINATION NUMBER to the top right-hand side of the page.

    9.5 Insert your question number to the top left-hand side of the page.

    9.6 Print the page.

10. You are allowed to make use of the help files in VB.NET.

11. The following files will appear on your hard drive in the folders:

    11.1 QUESTION 1
    … \PCP L3 P2 SUP 2018\TaxThandi\ TaxThandi.sln

    11.2 QUESTION 2
    … \PCP L3 P2 SUP 2018\YearMarks\YearMarks.sln

    11.3 QUESTION 3
    … No files

Copyright reserved

QUESTION 1

The following questions refer to the VB.NET solution called *TaxThandi (… \PCP L3 P2 SUP 2018)* that has been saved on the hard drive.

The lecturer of the Accounting subject at your TVET college has approached you to design an application that will assist the financial accounting students to understand how income tax in South Africa is calculated. Income tax is a tax that the government imposes on the financial income generated by an individual.

In order to determine how much tax an individual should pay, the person's taxable income should first be determined.

According to law, tax rates are determined by the Minister of Finance and fixed or passed by Parliament every year. Normal tax tables for the current tax year are shown in TABLE 1.1 below.

TABLE 1.1: RATES OF TAX INCOME PER GROUP

TAXABLE INCOME	KNOWN AS	TAX RATE
R0 up to R189 880	Lower brackets	18%
R189 880 up to R296 540	Middle brackets	26%
R296 541 up to R410 460	Upper brackets	31%
R410 461 and above	Others	45%

An individual also qualifies for a rebate which is an amount that is deducted from the normal tax payable by each income taxpayer. The amount that will be deducted is determined by the age group of the individual tax payer, as shown in Table 1.2 below.

TABLE 1.2: TAX REBATES

TYPE	CRITERION	AMOUNT TO BE DEDUCTED
Primary	under 65 years	R13 635
Secondary	65–75 years	R7 479 + Primary
Tertiary	over 75 years	R2 493 + Primary + Secondary

If the rebate amount is greater than normal tax, then the annual tax is zero.

The application will take the individual's income as an input and determine the following:

(a) The annual taxable income
(b) Applicable tax rate as per TABLE 1 above
(c) Total rebate to be deducted
(d) Annual tax (for 12 months)
(e) Monthly tax

Copyright reserved

Below is the IPO diagram for the problem:

INPUT	PROCESSING	OUTPUT
Person's income	Taxable income = monthly income * no. of months	Taxable income
	Annual tax = (taxable income * tax rate)	Rebate
Person's age group	IF Rebate > annual tax THEN annual tax = 0 ELSE Annual tax = (taxable income * tax rate) - rebate	Tax rate Annual tax
	Monthly tax = annual tax/12	Monthly tax

FIGURE 1 below is a screen shot of the interface.

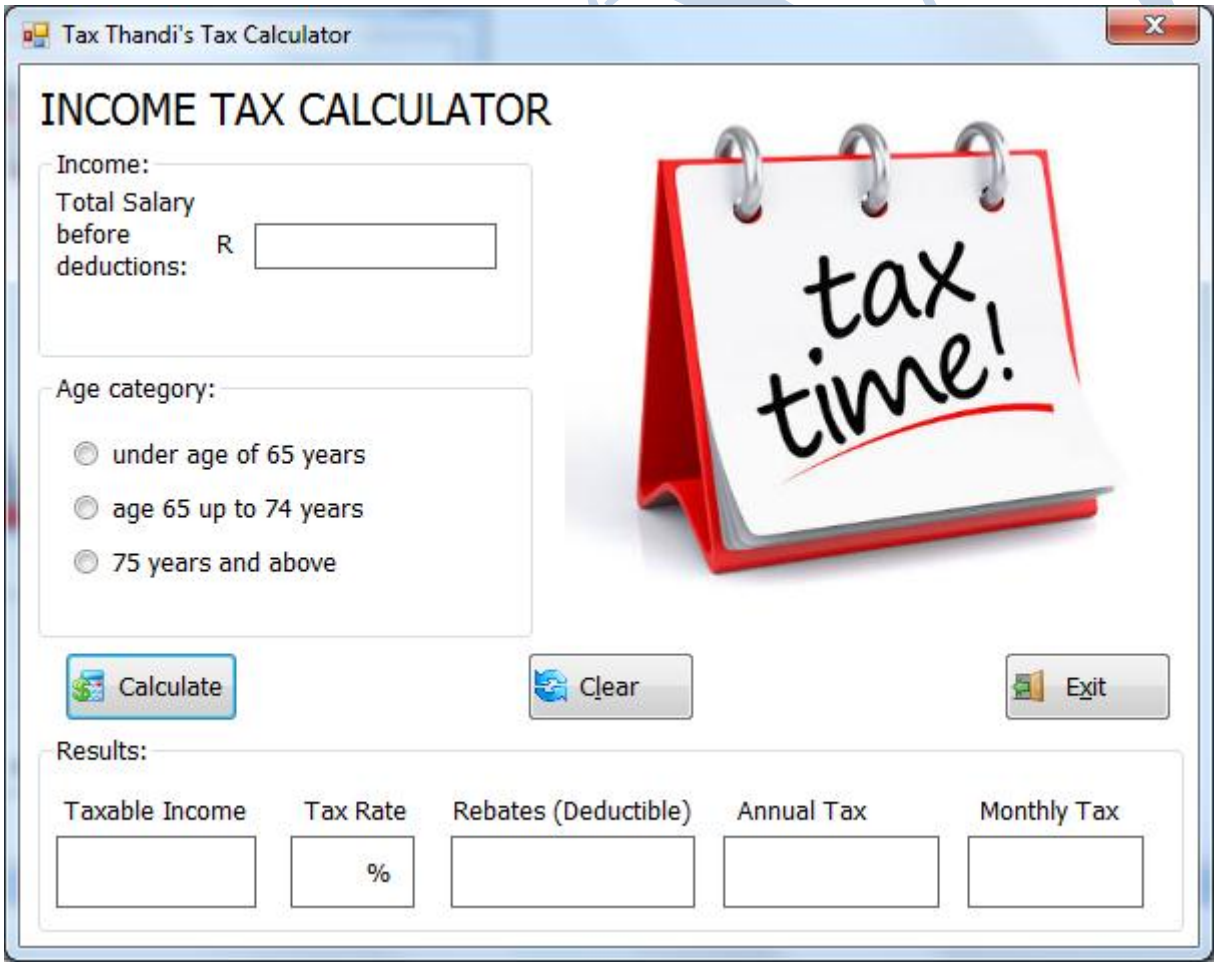
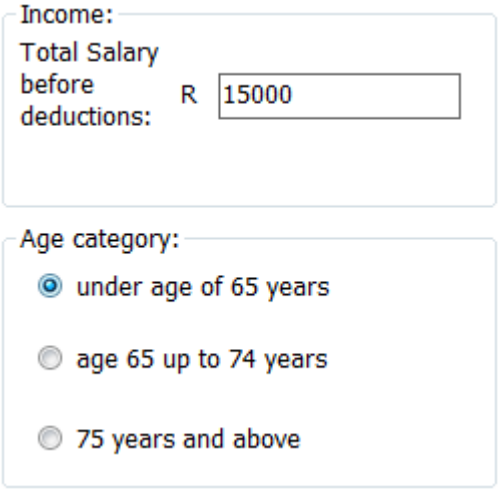


FIGURE 1

Copyright reserved

The following steps explain how to use the Income Tax Calculator application to calculate an individual person's annual tax payable.

STEP 1 RUN the application, enter the monthly income and select a taxpayer's age category.



STEP 2 Click the *Calculate* button. If the input data is NOT numbers, then an error message will be displayed exactly as shown below.

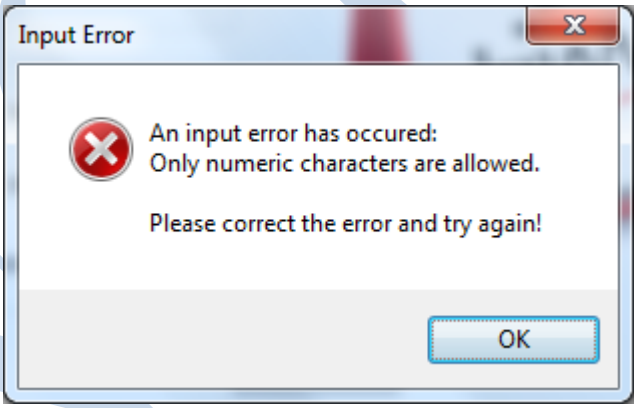
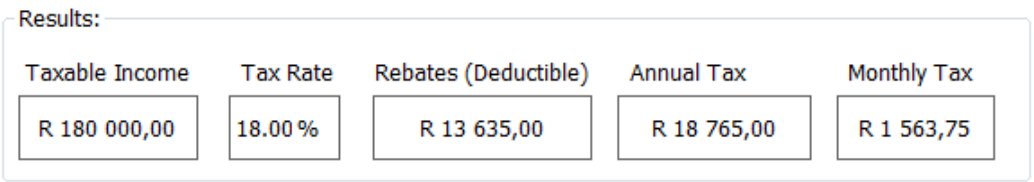


FIGURE 2

STEP 3 If the user enters a valid monthly income and selects a taxpayer's age category and clicks the C*alculate* button, then the application will determine the income tax amounts.



Copyright reserved

The completed interface with partially completed source code has been saved on your computer. Open the solution *TaxThandi* and follow the instructions to complete the application:

1.1 Declare the following at *form level*:

1.1.1 All appropriate variables (1)

1.1.2 An appropriate constant to store the number of months in a year, and THREE constants to store the values of each *rebate amount* respectively. (2)

1.2 When the form loads:

1.2.1 All radio buttons on the interface must be deselected.

1.2.2 Highlight the *Monthly Income* textbox background in yellow.

(2 x 1) (2)

1.3 Create a subprocedure called *ValidateInput* with the following *value parameters*:

- sender As System.Object
- e As System.EventArgs

The procedure needs to perform the following:

1.3.1 The subprocedure must *handle* the *LostFocus* event of the *Monthly Income* textbox. (2)

1.3.2 The subprocedure must test to ensure that the *Monthly Income* textbox contains numbers only. If NOT, the program must alert the user with a message as shown in STEP 2, FIGURE 2 and move the cursor to the appropriate control. (5)

1.3.3 Another option is to assign the value of the *Monthly Income* textbox to the appropriate variable. (1)

1.4 Create a user defined subprocedure called *CalcTaxableIncome* without parameters to calculate the amount that is taxable for the year. (2)

1.5 Create a user defined subprocedure called *DetermineRebateAmount* without parameters to determine the *rebates* applicable, based on the taxpayer's age group. See TABLE 1.2 above. (4)

1.6 Create a user defined subprocedure called *DetermineTaxRate* with ONE parameter *TaxableIncome*, to determine the appropriate rate of tax for the given *TaxableIncome*. See TABLE 1.1 above. (4)

Copyright reserved

1.7 Create a user defined function procedure called *CalcAnnualTax* and add the following code to the procedure.

1.7.1 Add a *Try … Catch* block to handle any exceptions that may occur when calculating the annual tax. When any exceptions occur, alert the user appropriately. (2)

1.7.2 The function procedure must calculate the *Annual Tax* amount. If the rebate amount is greater than normal tax, then the annual tax is zero. (6)

1.8 Add the following code to the *Click Event* of the *Calculate* button:

1.8.1 Validate that all the *Monthly Income* textbox is NOT empty. If empty alert the user, move the cursor to the appropriate control, and terminate the operation. (2)

1.8.2 Validate that the user has selected an *Age Category*. If NOT alert the user, and terminate the operation. (2)

1.8.3 Invoke the *CalcTaxableIncome*, *DetermineTaxRate* and *DetermineRebateAmount* sub procedures. (1)

1.8.4 Invoke the user defined function procedure *CalculateAnnualTax* to calculate the *Annual Tax*. (1)

1.8.5 Determine the *Monthly Tax* that the individual will have to pay. (1)

1.8.6 Invoke the *DisplayResults* subprocedure. (1)

1.9 Insert your EXAMINATION NUMBER in the source code and make a printout of the source code.

1.10 Change the text in the *title bar* of the *form* to your EXAMINATION NUMBER.

Run the program, enter the following test data, click the confirm order button and make a screen print of the output.

Monthly income: R15 000
Age : 24 (1)

[40]

QUESTION 2

The following questions refer to the VB.NET solution *YearMarks (… \PCP L3 P2 SUP 2018\YearMarks)* that has been saved on the hard drive.

You have been approached by the MIS manager of Tshwane Technical College who wants an application that would allow the subject lecturers to calculate a students' year mark after they have written all their assessments. The application will capture the students details together with the marks obtained in each assessment and then determine the year mark as well as the average year mark for the class.

All courses at the college have four assessments that are conducted at various times during the year. Two of the assessments are formative assessments that are referred to as *Tests* and the other two are practical assessments that are referred to as *Assignments*. Each type of assessment contributes a different percentage of marks to the year mark which is indicated below in TABLE 2.

TABLE 2: YEAR MARK ASSESSMENT WEIGHTING

NO.	ASSESSMENT	CONTRIBUTION TO YEAR MARK
1	Test 1	20%
2	Test 2	20%
3	Assignment 1	30%
4	Assignment 2	30%
	TOTAL	100%

The application uses THREE one-dimensional arrays and ONE two-dimensional array in parallel, to store all the information.

FIGURE 3 below is a print screen of what the user interface looks like.

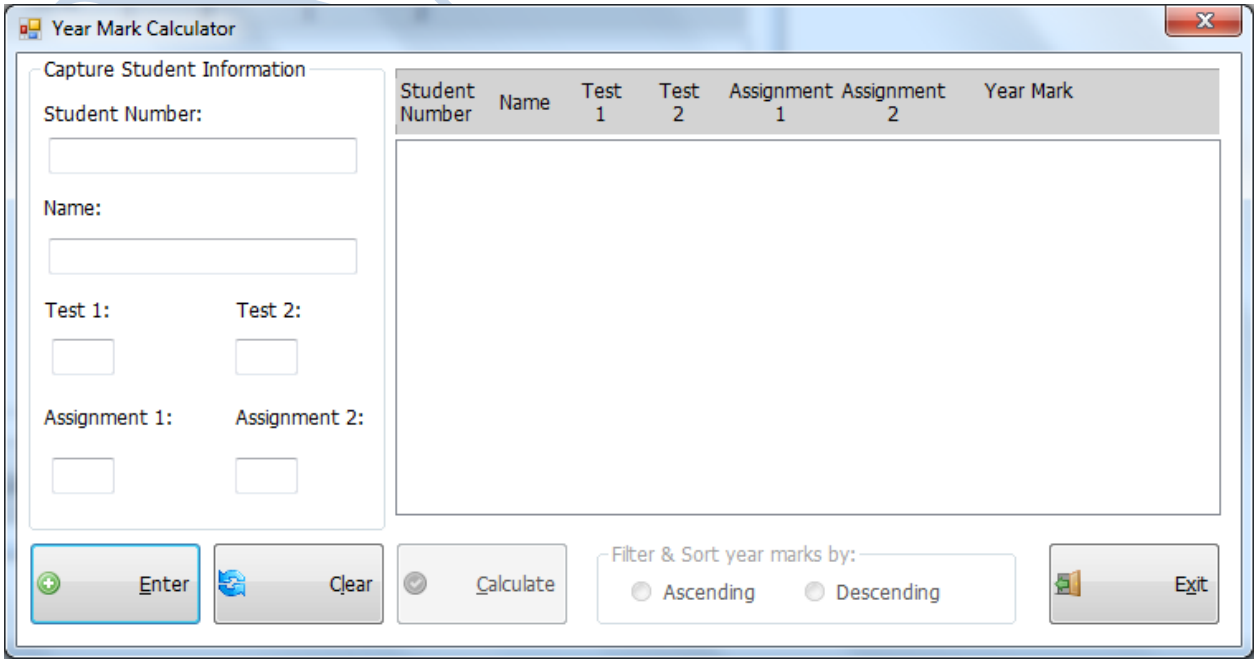
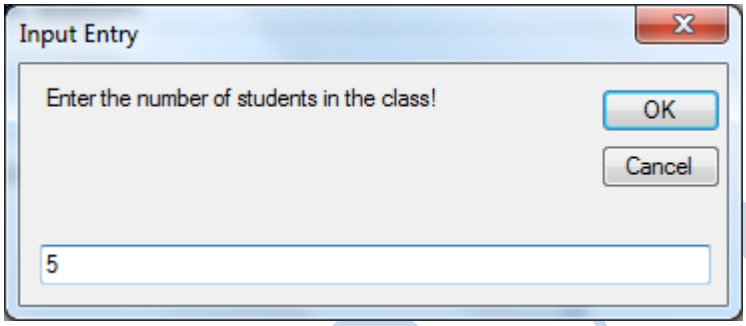


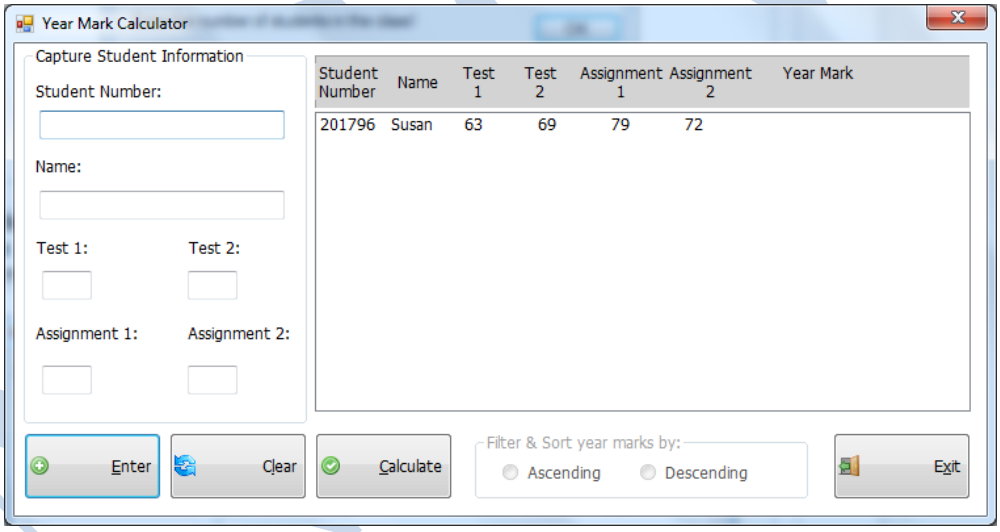
FIGURE 3

The following steps explain how to use the program to enter and calculate the year marks for students.

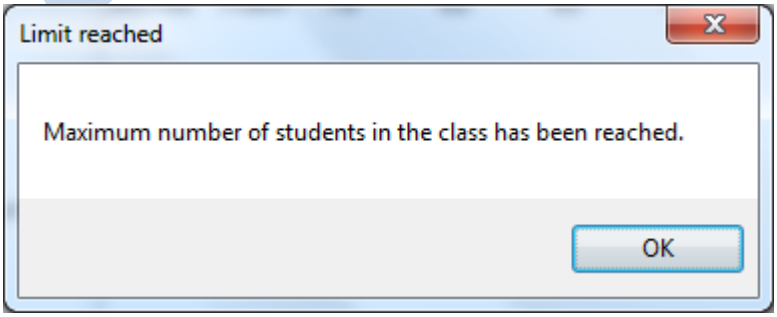
STEP 1 Run the application. The user will be prompted by an input dialogue box, to enter the number of students in the class. Enter the total number of students in the class and click the OK button.



STEP 2 Enter the student's student number, name and assessment marks. Click on the *Enter* button to add the information to the arrays. After each entry the current student's entry information will be added to the list box.

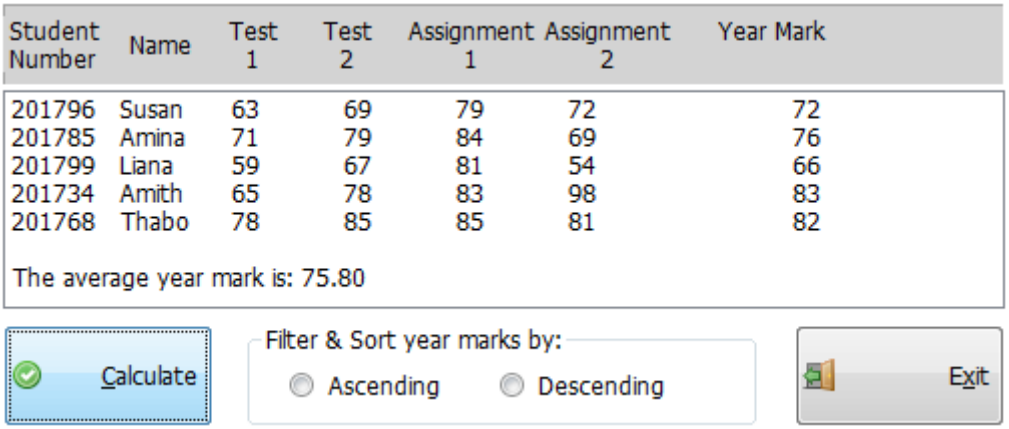


STEP 3 If the user attempts to enter more students than initially specified, the application will alert the user with a message dialogue box.



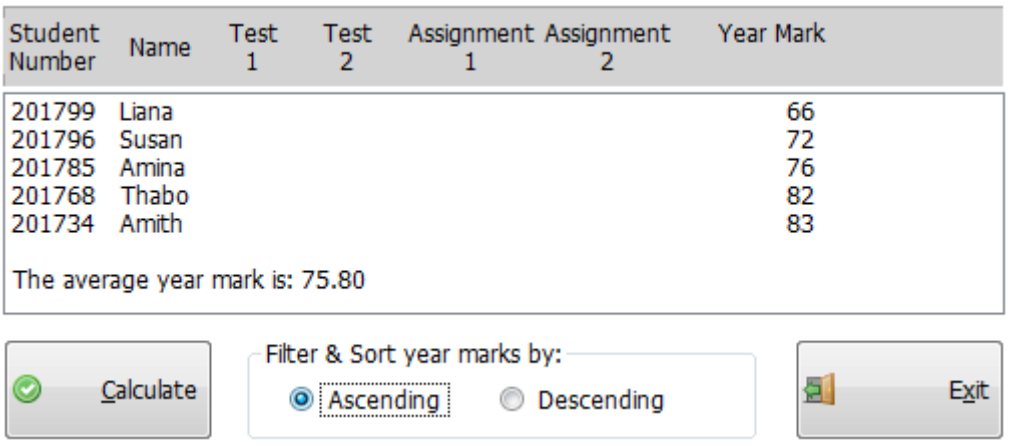
Copyright reserved

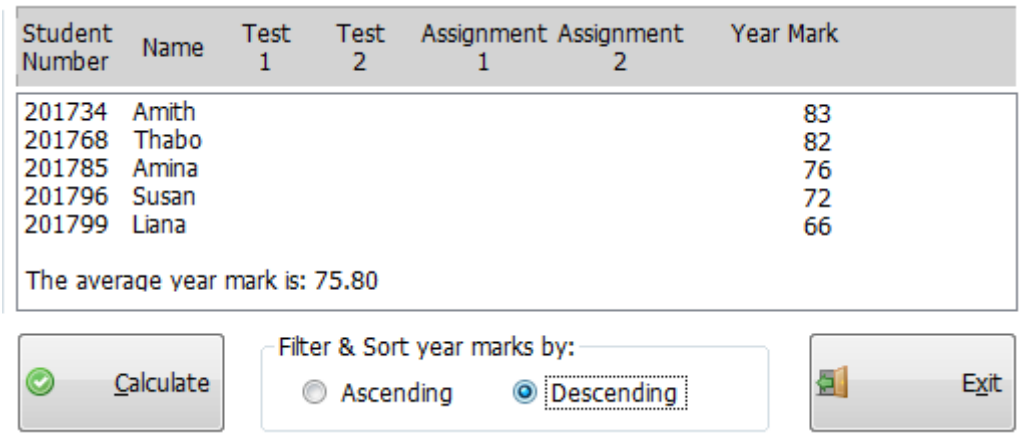
STEP 4 Click on the *Calculate* button to calculate the year mark for each student and to calculate the average year mark for the entire class.



STEP 5 The year marks can also be filtered and sorted according to the year marks either in ascending or descending order, by clicking the relevant radio button.

NOTE: Only the student number, name and year marks will be sorted and displayed.





The completed interface with partially completed source code has been saved on the hard drive. Open the solution *YearMarks* and follow the instructions to complete the program:

2.1 Add the following code to the *form load* event:

2.1.1 An input dialogue box must be presented prompting the user to enter the number of students in the class. The resultant number of students must then be assigned to a variable.

2.1.2 Change the size of all the arrays to the new size of the number of students.

(2 × 2) (4)

2.2 Create a user defined subprocedure called *GenerateYearMark* with NO parameters, to iterate through the *year marks* and generate the year mark based on the assessment's weightings specified in TABLE 2. (6)

2.3 Create a user defined-function procedure called *CalcYearMarkAverage* with NO parameters, to calculate and return the *Average Year Mark* of all the students. See STEP 4. (7)

2.4 Add the following program code to the click event of the *Enter* button:

2.4.1 The application must test to ensure that the number of students being entered is not more than the number of students initially specified in QUESTION 2.1.1. If more, alert the user, invoke the *LimitReached* subprocedure and terminate the operation.

HINT: Make use of the counter variable. (4)

2.4.2 Assign the values from the variables into the arrays.

HINT: Make use of the counter variables, rows and columns to add data to the two-dimensional array. (2)

2.4.3 Display the student number, student name and assessment marks in the list box.

HINT: When displaying the TWO-dimensional array, set the columns manually. (1)

2.5 Add the following program code to the click event of the *Calculate* button.

2.5.1 Invoke the *GenerateYearMark* function procedure to calculate the year mark for the class. See STEP 4 above. (1)

Copyright reserved

2.5.2 Iterate through the arrays to display the student number, student name and assessment marks in the listbox. (2)

2.5.3 Invoke the *CalcYearMarkAverage* function procedure to calculate the average year mark for the class. See STEP 4 above. (1)

2.6 Add the following program code to the check changed event of the *Ascending Radio* button.

2.6.1 Sort the elements in the *student number*, *student name* and *year mark* arrays, according to *year marks* in ascending order, by using a *Bubble Sort* algorithm. (4)

2.6.2 Display the results in the listbox. See STEP 5 above. (1)

2.7 Add the following program code to the check changed event of the *Descending Radio* button.

2.7.1 Sort the elements in the *student number*, *student name* and *year mark* arrays, according to *year marks* in descending order, by using a *Bubble Sort* algorithm.

2.7.2 Display the results in the listbox. See STEP 5 above.

(2 × 1) (2)

2.8 Add your EXAMINATION NUMBER to the source code. Make a printout of the source code.

[35]

QUESTION 3

The senior administration officer at Mpumalanga Technical College manages the college's logbook for the use of college vehicles. Currently the college uses manual forms for the booking of vehicles for college activity-related trips. This manual system is becoming too difficult for the administration officer and you have been asked to design a database where the senior administration officer can record and keep track of all the booking information of vehicle usage.

Use the information below to set up the database tables in MS Access. Open a new blank database and save it as *Vehicles* with your EXAMINATION NUMBER, e.g. Vehicles – EXAMINATION NUMBER.

The database design has been normalised to 3NF and consists of TWO data tables (Vehicle, Employee) and ONE link table (Booking).

Copyright reserved

Create the Vehicle table and save it as tblVehicle.

FIELD NAME	DATA TYPE	FIELD SIZE	DESCRIPTION
vehRegistrationNo	Text	8	Primary key (unique code)
vehMake	Text	15	Manufacturer of the vehicle
vehModel	Text	30	Model name of vehicle
vehEngineCapacity	Text	4	Engine capacity
vehMaxPassenger	Number	Byte	Number of passengers

Create the Employee table and save it as tblEmployee.

FIELD NAME	DATA TYPE	FIELD SIZE	DESCRIPTION
empIdentityNo	Text	13	Primary key (unique identifier)
empFullName	Text	45	Name and surname
empLicenseCode	Text	3	Driver's license code

Create the Booking table and save it as lnkBooking.

FIELD NAME	DATA TYPE	FIELD SIZE	VALIDATION RULE	DESCRIPTION
lnkEmpIdentityNo	Text	13		Primary key
lnkRegistrationNo	Text	8		Primary key
lnkBookingDate	Date	Short Date		Date of booking
lnkDestination	Text	30		Destination of trip
lnkDistance	Number	Integer	>30 & < 600	Estimated distance
lnkPassengers	Number	Byte		Number of passengers
lnkOdoReadOut	Number	Long integer		Odometer reading before trip
lnkOdoReadIn	Number	Long integer		Odometer reading after trip

NOTE: The college has the following policy that applies to the booking of a vehicle:

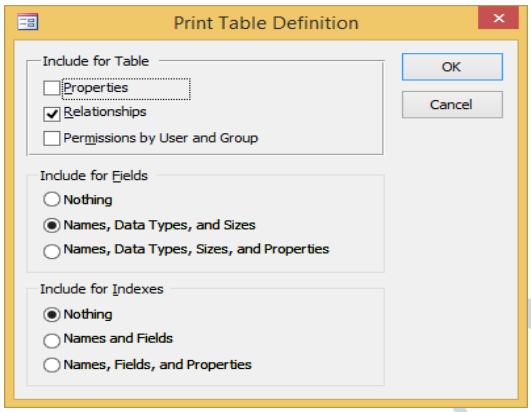
A vehicle can only be booked if the estimated distance being travelled is more than 30 kilometres but less than 600 kilometres.

Create a One-to-Many relationship between the EMPLOYEE and BOOKING table. Use the employee identity number field and enforce referential integrity.

Create a One-to-Many relationship between the VEHICLE and BOOKING table. Use the vehicles registration number field and enforce referential integrity.

Copyright reserved

3.1 Set the following options for ALL THREE tables using the database documenter.



Print the table definition for ALL THREE tables. (8)

3.2 Open the database table tblVehicle and add the following records:

REGISTRATION NO.	MAKE	MODEL	ENGINE CAPACITY	MAXIMUM PASSENGERS
FE13RTGP	VW	Golf	1.6	5
WTR102GP	Nissan	Hardbody	2.0	3
ZDT995GP	VW	Polo	1.4	5
ZPP880GP	Toyota	Ses'fikile	2.4	16

3.3 Open the database table tblEmployee and add the following records:

EMPLOYEE IDENTITY NO	EMPLOYEE NAME	LICENSE CODE
8001315449081	Ms SE Tsebe	EB
8203195009082	Mr G van Tonder	EB
7604211225084	Mr HK Ramagoshi	C1

3.4 Open the database table lnkBooking in *Design View* and perform the following instruction:

Select the *lnkDistance* field and make a screen print of the table's design. (1)

3.5 The capturing of the booking data should be done as follows:

Refer to FIGURE 4 below.

3.5.1 Use the form wizard to create a subform (a form within a form) for the *Employee* and *Booking* tables.

3.5.2 Change the title to Vehicle Requisition – EXAMINATION NUMBER and save the form as frmVehicleRequistion.

Copyright reserved

3.5.3 Make a print screen of the form. Ensure that ALL columns are clearly visible.

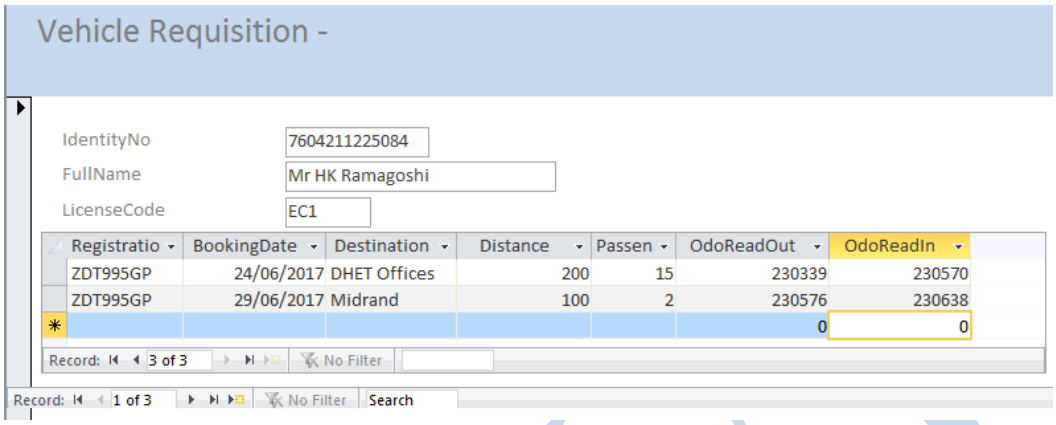


FIGURE 4

(3 × 1) (3)

3.6 Enter the booking data for the following employees:

Ms SE Tsebe

Registration number	Date	Destination	Estimated distance	Passengers	Odometer reading Out	Odometer reading In
FE13RTGP	2017-06-21	Pretoria	200	2	59660	59870
WTR102GP	2017-06-29	Midrand	100	2	132045	132107
FE13RTGP	2017-07-25	Centurion	60	4	59877	59989

Mr G van Tonder

Registration number	Date	Destination	Estimated distance	Passengers	Odometer reading Out	Odometer reading In
ZPP880GP	2017-06-21	Johannesburg	100	1	152090	152145

Mr HK Ramagoshi

Registration number	Date	Destination	Estimated distance	Passengers	Odometer reading Out	Odometer reading In
ZDT995GP	2017-06-24	DHET Offices	200	15	230339	230570
ZDT995GP	2017-06-29	Midrand	100	2	230576	230638

Copyright reserved

3.7 Create a database query using ALL THREE tables according to the following:

The query must retrieve the following fields:

EMPLOYEE table: Employee name
VEHICLE table: Model
BOOKING table: Registration number, Booking date, Destination, Distance

Add a calculated field called *Travelled kilometres*, to determine the actual distance travelled per trip, in kilometres.

Travelled kilometres are the difference between the Odometer Reading In and Odometer Reading Out.

Save the query as qryVehicleUsage with your EXAMINATION NUMBER.

3.7.1 Open the query in design view and make a screen print of the query design. Ensure that ALL columns are clearly visible. (7)

3.7.2 Run the query and print the resulting records on ONE page. (1)

3.8 Create the following database report using the query created in QUESTION 3.7.

3.8.1 The report must display all the fields from the query. (1)

3.8.2 View the report by Employee. (1)

3.8.3 Group the report by Vehicle model. (1)

3.8.4 Calculate the variance between the estimated distance and the actual distance travelled. (2)

Change the report header to VEHICLE USAGE - EXAMINATION NUMBER.

Save the report as rptVehicleUsage.

Make a printout of the report using LANDSCAPE orientation.

[25]

TOTAL: 100

Copyright reserved