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)

23 November 2018 (X-Paper)

09:00–13:00

This question paper consists of 18 pages.

Copyright reserved

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.

INSTRUCTIONS TO CANDIDATES AND INVIGILATORS

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 application software.
2. Complete these questions on a computer using MS Office 2003 or a later version and Visual Studio.NET 2005 or VB.NET 2005 Express or a later version.
3. Candidates must print their work immediately after they complete a question.
4. Candidates' EXAMINATION NUMBER must be on ALL PRINTOUTs.
   NO questions without a printed EXAMINATION NUMBER will be marked.
   PRINTOUTs with handwritten examination numbers will NOT be marked.
5. In the event of a power failure, computer breakdown or a printing device breakdown, the invigilator will make the necessary arrangements to continue with the examination session once the problem has been resolved. Candidates will not be penalised for time lost under these circumstances.
6. Candidates must save ALL work on the computer at regular intervals to prevent any loss of work. NO additional time may be granted for any work lost due to lack of saving.

Copyright reserved

7. NO candidate may print any work for another candidate.
NO candidate may make a secondary storage device available to another candidate.
NO candidate may 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.

8. Candidates MUST hand in ALL PRINTOUTS of the source code and screen PRINTOUTs done during the examination session.

9. Candidates may NOT 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.

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

    10.1 Run/Debug the solution.

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

    10.3 Open MS Word document and paste the screen dump.

    10.4 Insert your EXAMINATION NUMBER and QUESTION NUMBER in a header against the left margin.

    10.5 Print the page.

11. You may use of the help files in VB.NET.

12. The following files will appear on the computer in the folders indicated below:

    12.1 QUESTION 1
    … \PCP L3 P2 SET A OF 2017\HomeLoan

    12.2 QUESTION 2
    … \PCP L3 P2 SET A OF 2017\ExamMarking

    12.3 QUESTION 3
    … No files

Copyright reserved

QUESTION 1

The following questions refer to the VB.NET solution called HomeLoan (… \PCP L3 P2 SET A OF 2017\HomeLoan) that has been saved on the computer.

A local home loan originator has requested you to design an application to give their customers an approximate idea of the costs involved in purchasing a house by using a home loan. The application will be installed on kiosk computers at all their branches so that the customers can use the application to determine how much they would approximately pay to purchase their own home using a fixed interest rate.

The application takes as input the principal amount which will be borrowed from the bank, the annual interest rate, which can be the prime lending rate of 10,25 or higher, and the repayment period in years. The application then estimates the monthly repayments for the home loan, the total amount paid back at the end of the repayment period as well as the total amount paid back as interest.

FIGURE 1 below is a print screen of the graphical user interface.

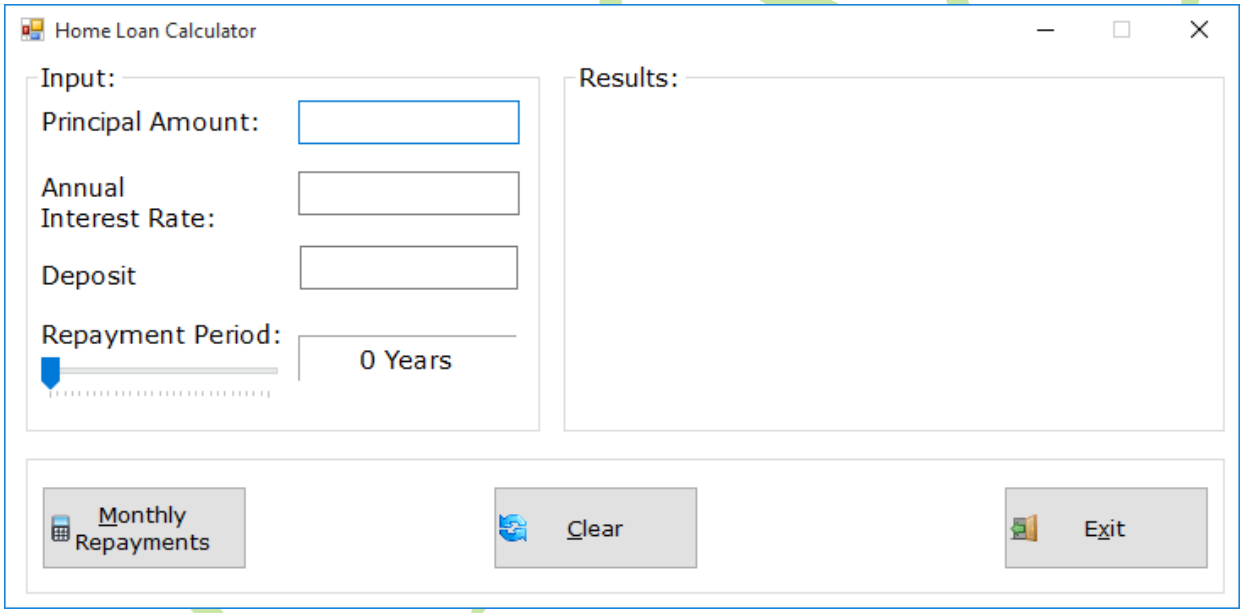


FIGURE 1

Below is the IPO diagram for the problem:

INPUT	PROCESSING	OUTPUT
Principal Amount Interest Rate Repayment period (years)	Principal Amount = Principal Amount – Deposit Annual Interest Rate = Interest Rate/100 Number of Months = Repayment period/12 Monthly Interest Rate = Annual Interest Rate/12 Monthly Instalment = -Pmt(Monthly Interest Rate, Number of Months, Principal Amount) Total Amount = Monthly Instalment * Number of Months payable Total Interest = Total Amount Payable – Principal Amount payable	Monthly Instalment Total Amount Payable Total Interest Payable

NOTE: Pmt is a built-in financial function which calculates the repayment for a loan based on constant payments and a constant interest rate.

The following STEPS explain how to use the HomeLoan application.

STEP 1 Run the application and capture the principal amount.



If the user enters text as the principal amount, then the application should display an error message as illustrated in FIGURE 2.1 below.

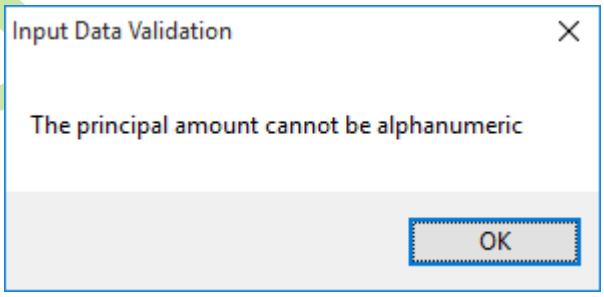


FIGURE 2.1

Copyright reserved

STEP 2 Enter the interest rate.



If the user enters an interest rate lower than the current prime lending rate, then the application should display an error message as illustrated in FIGURE 2.2 below.

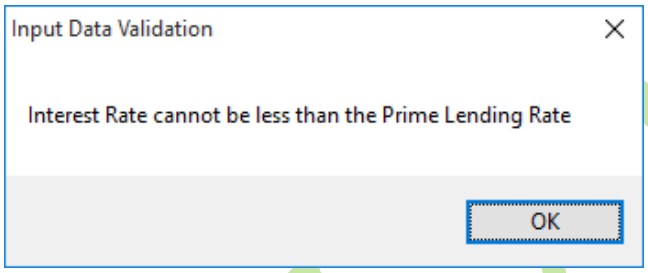
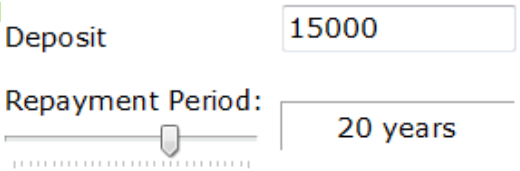


FIGURE 2.2

STEP 3 Enter the deposit amount if provided and use the track bar control to select an appropriate repayment period.



If the user does not select a repayment period, then the application should display an error message as illustrated in FIGURE 2.3 below.

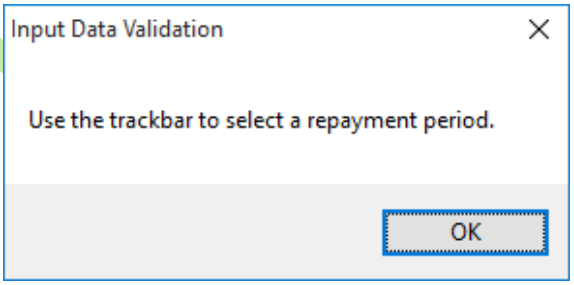


FIGURE 2.3

Copyright reserved

STEP 4 Click on the monthly repayments button. The application should output results as shown in FIGURE 2.4 below.

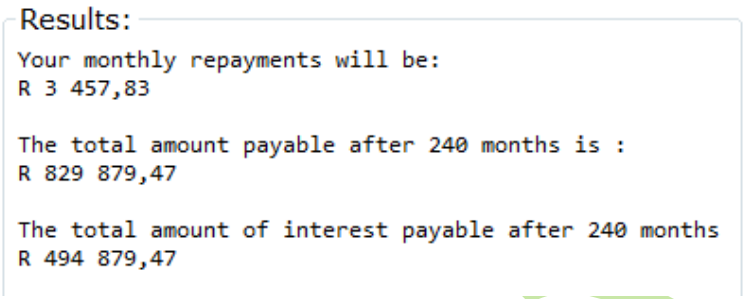


FIGURE 2.4

The completed interface has been saved on the computer. Open the VB.NET solution HomeLoan and follow the instructions to complete the application.

Add your EXAMINATION NUMBER as a comment to the source code.

1.1 Declare and Initialise THREE appropriate *form level* variables to which the *input* will be assigned.

HINT: Use an 8-byte *precision floating point* data type for storing currency related amounts. See IPO diagram. (2)

1.2 Create a user defined subprocedure called *GetInput*, using a restricted access modifier which will perform the following:

1.2.1 The subprocedure declaration must receive THREE parameters in a way which will allow the procedure to change the values of the THREE *input* variables. (4)

1.2.2 Calculate the *principal amount*, *annual interest rate* and *number of months* and assign the values to the respective variables. (5)

1.3 Create a user defined subprocedure called *Repayment*, using a restricted access modifier, which will perform the following:

1.3.1 The subprocedure declaration must receive as copies the *principal amount*, the *annual interest rate* and *repayment period*. (4)

1.3.2 Declare appropriate variables to which will be assigned the *monthly interest rate*, *monthly instalment*, *total amount payable* and *total interest payable*. (2)

1.3.3 Calculate the *monthly interest rate*, *monthly instalment*, *total amount payable* and *total interest payable* and assign the values to appropriate variables. (8)

Copyright reserved

1.3.4 Clear the list box and then display the *monthly interest rate*, *monthly instalment*, *total amount payable* and *total interest payable*, formatted as currency with two decimal places.

SEE STEP 4, FIGURE 2.4 above. (3)

1.4 In the *scroll event* of the *Repayment period* track bar:

1.4.1 Assign the *value* of the track bar together with an appropriate message to the repayment period label.

HINT: See STEP 3. (1)

1.5 In the *click event* of the *Calculate Repayment* button:

1.5.1 The application should perform validation as follows:

(a) The text entered as the principal amount should be a number only.

(b) The interest rate entered cannot be less than the prime lending rate in South Africa.

(c) Check that a repayment period has been selected.

If the validation criteria are not satisfied, then display an appropriate message, move the cursor to the relevant control and abort the procedure. (3)

1.5.2 If the deposit text box is empty, then assign a value of zero to the deposit text box. (1)

1.5.3 Invoke the *GetInput* and *Repayment* procedures respectively. (4)

1.6 When the *Clear* button is clicked:

1.6.1 Use a method to erase the contents of the list box and text box controls.

NOTE: Do NOT set the *Text* property of the control(s) to **" "**

1.6.2 Reset the repayment period label and the track bar controls to zero.

(2 × 1) (2)

1.7 Make a PRINTOUT of the source code.

Copyright reserved

1.8 Change the text in the *title bar* of the *form* to your examination number.

1.8.1 Run the application, enter the following test data, click the Calculate Repayment button and make a screen print of the output.

Amount: 500 000
Interest Rate: 10,5
Repayment Period: 25 years (1)

[40]

QUESTION 2

The following questions refer to the VB.NET solution called ExamMarking (… \PCP L3 P2 SET A OF 2017\ExamMarking) that has been saved on the computer.

The Department of Basic Education is about to embark on the National Senior Certificate examination, and a team of markers have asked you to design an application to assist them when marking a first additional language examination paper. The question paper (second paper) consists of FOUR sections. Each section is worth 35 marks. Students must answer questions from TWO sections ONLY. Meaning that the exam paper is valued at 70 marks.

If questions from more than two sections are answered, then the marks from the first two sections only are recorded, whilst latter ones are ignored and an appropriate message is then displayed. Unanswered sections are indicated by a zero mark.

FIGURE 3 below is a print screen of the graphical user interface.

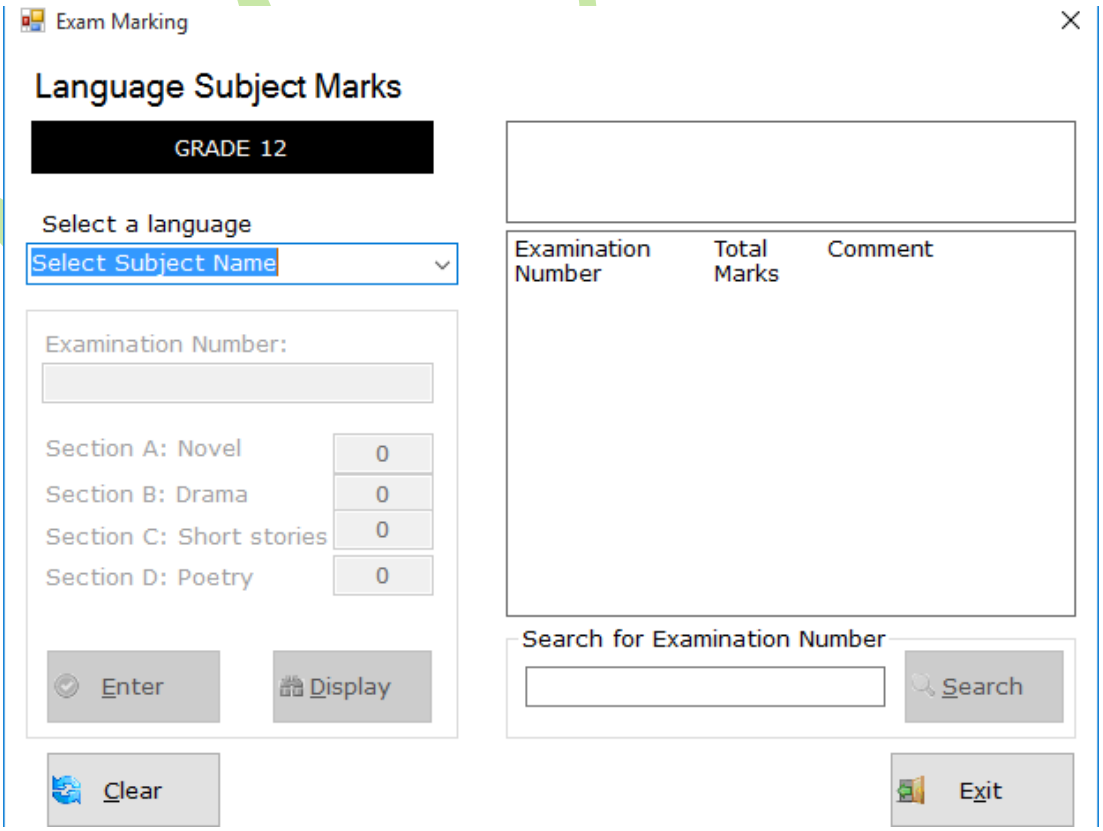
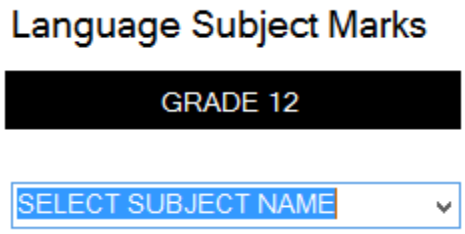


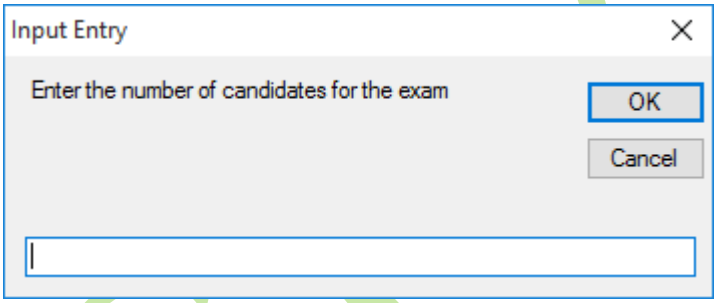
FIGURE 3

The following STEPS explain how to use the ExamMarking application to enter marks for several students and display their final marks:

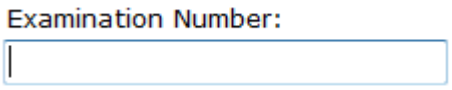
STEP 1 Run the application and select the subject name.



STEP 2 Enter the number of candidates that wrote the examination.



STEP 3 Enter the examination number of the candidate.



If the user omits the examination number, then the application should display an error message as illustrated in FIGURE 4.1 below.

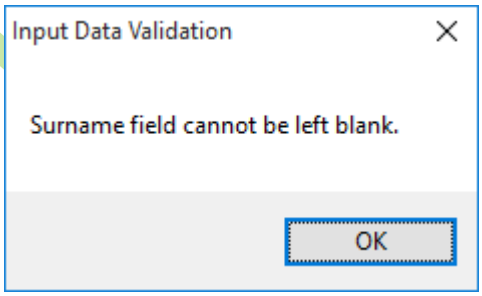
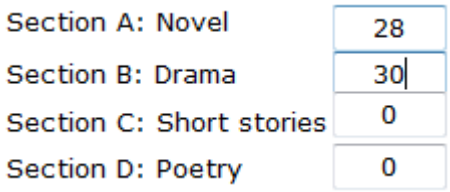


FIGURE 4.1

STEP 4 Capture all marks obtained for sections answered. Unanswered sections are indicated by a zero mark.



If the user enters a mark greater than 35, then the application should display an error message as illustrated in FIGURE 4.2 below.

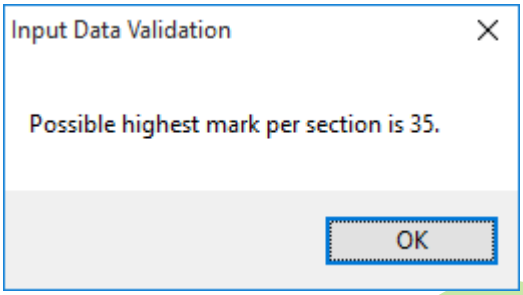


FIGURE 4.2

STEP 5 Click the Enter button to add the information to the arrays.



Capture the rest of the information for all the candidates (STEPS 1 to 4), and after each entry click the next button.

If the user attempts to enter more candidates than the number of candidates indicated in STEP 2 the application should display an error message as illustrated in FIGURE 4.3 below.

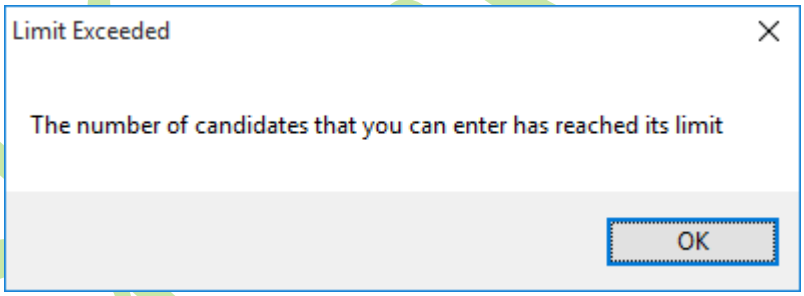


FIGURE 4.3

If the student answered questions from more than TWO sections, then the first two marks are counted, and an appropriate comment is displayed next to the final mark as illustrated in FIGURE 4.4 below.

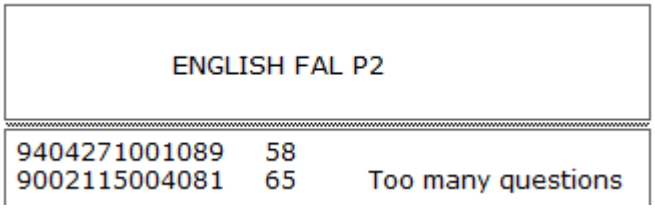
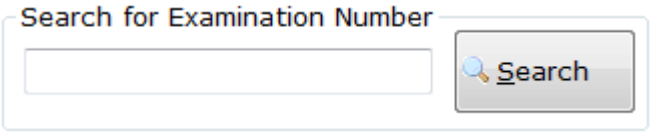


FIGURE 4.4

STEP 6 If the user wants to search for an examination number, enter the examination number in the text box and click on the search button.



If the application finds the examination number being searched for the results must be displayed in the list box as illustrated in FIGURE 4.5 below.

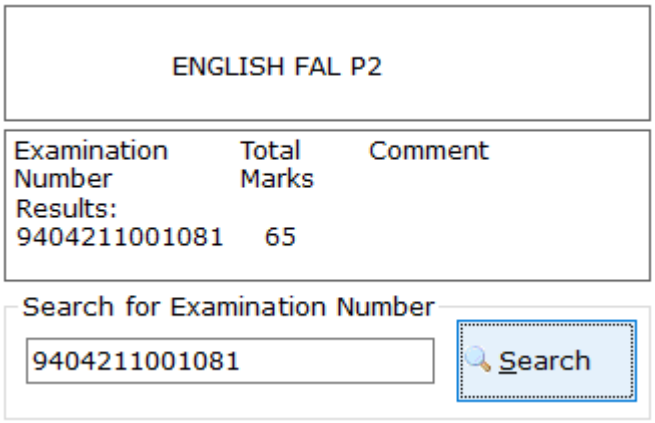


FIGURE 4.5

If the application does not find an examination number, an appropriate message must be displayed as illustrated in FIGURE 4.6 below.

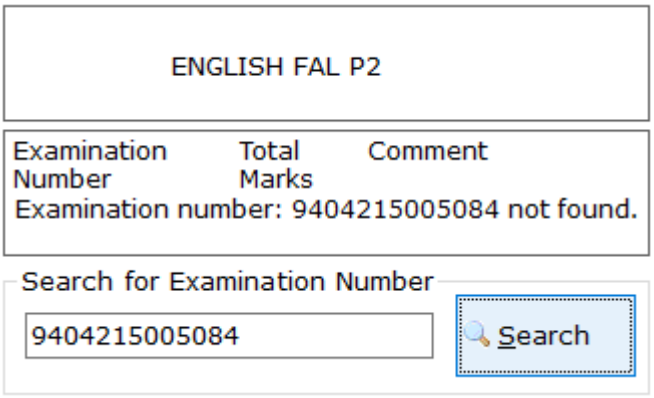


FIGURE 4.6

The completed interface with partially completed source code has been saved on the computer. Open the VB.NET solution ExamMarking and follow the instructions to complete the application.

Copyright reserved

Add your EXAMINATION NUMBER to the source code.

2.1 Declare a one-dimensional string array called *strArrSubjectNames* and initialise it with the following values:

'ENGLISH FAL P2' and 'SEPEDI FAL P2'. (1)

2.2 Declare THREE appropriate global variables as described below:

- A counter variable to be used when looping through the arrays
- A variable that will hold the number of sections answered
- A variable that will hold the number of candidates for the exam (1)

2.3 In form *load* event:

2.3.1 Use a `<Do Until ... Loop>` to populate the *Subject* combo box with the values from the *strSubjectNames* array. (4)

2.4 When the content of the *combo* box changes:

2.4.1 An *input* dialog box must be presented, prompting the user to enter the number of candidates for the exam. The resultant number of candidates must then be assigned to a variable.

2.4.2 Change the size of all the arrays, except the array called *strArrSubjectNames*, to the number of candidates.

(2 × 2) (4)

2.5 Create a user defined subprocedure called *ValidateExamNo*. The procedure must determine if the examination number text box is blank. If so, alert the user with appropriate message, and move the cursor to the examination number textbox. (3)

2.6 Create a user defined function procedure called *ValidateMark* with ONE parameter called *value*. The function procedure must determine if the *value* is greater than 35. If so, alert the user with an appropriate message, and return FALSE. Otherwise, return TRUE. (4)

2.7 Add the following code to the click event of the *Enter* button.

2.7.1 The application must test to ensure that the number of candidates being entered is the same as the number of candidates initially specified using the input dialog box.

If NOT, alert the user with an appropriate message, invoke a subprocedure called *ClearInputData,* disable the *Enter* button and abort the procedure.

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

Copyright reserved

2.7.2 Assign the examination number(s) to the appropriate array. (1)

2.7.3 Make use of a structured <*IF…END IF*> statement to determine from which of the four sections marks were entered and assign those marks to their respective arrays.

NOTE: If marks from more than TWO sections are entered, the application will only capture marks from the first TWO sections and capture zero for the other TWO sections. (4)

2.7.4 Increment the *number of sections answered* variable. (1)

2.7.5 If more than TWO sections were answered, assign an appropriate message to the array; otherwise assign an empty string to the array.

See STEP 5, FIGURE 4.4 above. (2)

2.7.6 Reset the *number of sections answered*. (1)

2.8 Add the following code to the click event of the *Search* button.

2.8.1 Write appropriate code to search for an examination number and corresponding total marks only, and display the results in the list box.

See STEP 6 above. (4)

2.9 Add your EXAMINATION NUMBER to the source code. Make a PRINTOUT of the source code.

2.10 Run the application, enter the following test data and make a screen print of the output. (1)

Language: English FAL P2
No of Candidates: 2

Examination Number	**Novel**	**Drama**	**Short Stories**	**Poetry**
8306231006089	20		20	
8706126008084	12	12	13	

[35]

QUESTION 3

The CEO of a private college that offers various IT related short courses in your hometown has hired you to design a database application to store information about the services they offer and to generate various reports.

The CEO would like to store the courses they offer and their clients' basic information as well as which client enrolled for which courses. This would enable the CEO to draw monthly reports. One such report shows short course names, names of clients, minimum amount paid and outstanding balance per client. Included in the report are grand totals for each short course.

Open Microsoft Access. Create a new blank database and save it as Course together with your EXAMINATION NUMBER, e.g. Course-*EXAMINATION NUMBER.*

Use the information provided below to set up the database tables in MS Access. The database design consists of TWO data tables and ONE link table.

Create the Clients table as indicated below and save the table as tblCustomer.

Field Name	Data Type	Field size	Description
custNumber	Number	Integer	Primary Key
custFirstName	Text	25	Client's First Name
custLastName	Text	25	Client's Surname
custMobile	Text	10	Client's Mobile Number

Create the Courses table as indicated below and save the table as *tblCourse*.

Field Name	Data Type	Field size	Description
courseNumber	Number	Integer	Primary Key
courseName	Text	50	Short Course name
courseFee	Currency	2 decimal places	Course Fee

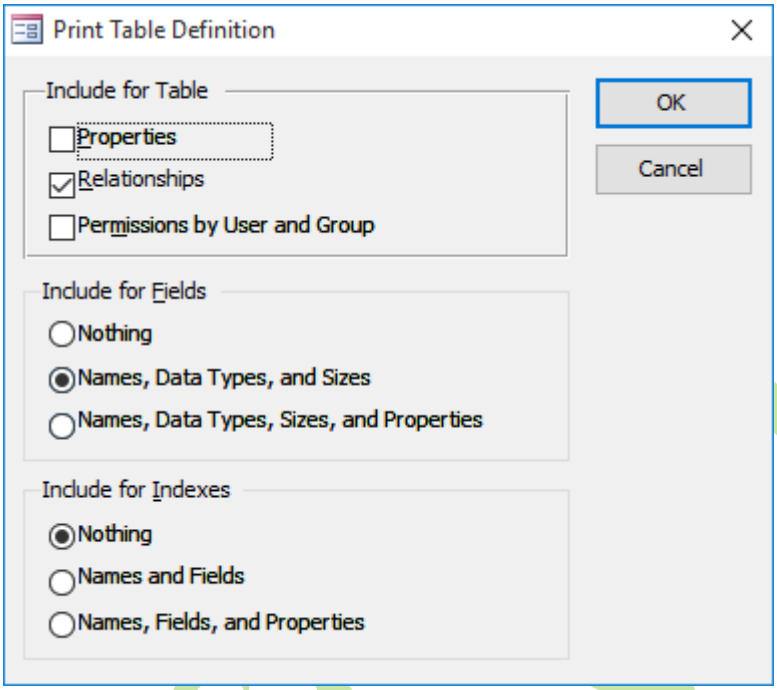
Create the Enrolments table as indicated below and save it as *lnkEnrolment*.

Field Name	Data Type	Field size	Description
custNumber	Number	Integer	Foreign Key
courseNumber	Number	Integer	Foreign Key
dateEnrolled	Date/Time	Long Date	Date Enrolled
amountPaid	Currency	2 decimal places	Initial payment

Create a 1:M relationship between *Customer* and *Enrolment*, as well as between *Course* and *Enrolment*, and enforce referential integrity.

Copyright reserved

3.1 Set the following options for all the tables using the database documenter.



Print the table definitions for all THREE tables. (7)

3.2 Capture the following customer and course details into the database.

3.2.1 Key in the following client records and make a PRINTOUT.

Cust Number	Cust FirstName	Cust LastName	Cell Phone
100	Maredi	MOLOTO	0610025540
101	Shane	SMITH	0832349908
102	Meriam	HELEN-AIDE	0726560885
104	Kingsley	KOLOBE	0731123349
105	Anna-Merie	VAN LINDEN	0842007783

3.2.2 Key in the following Course records and make a PRINTOUT.

Course Number	Course Name	Course Fee
24002	ICT in Education	R12 500,00
24003	C++ Programming	R14 000,00
24004	Web design	R9 200,00
24005	Networking fundamentals	R5 500,00
24006	Vb.net Programming	R10 500,00
24007	Introduction to Databases	R6 500,00

(2 × 1) (2)

Copyright reserved

3.3 Create a database form for capturing the enrolment as illustrated in FIGURE 5.1 below.

3.3.1 Create a form with subforms. Change the title to 'Customers' and save it as frmEnrollment.

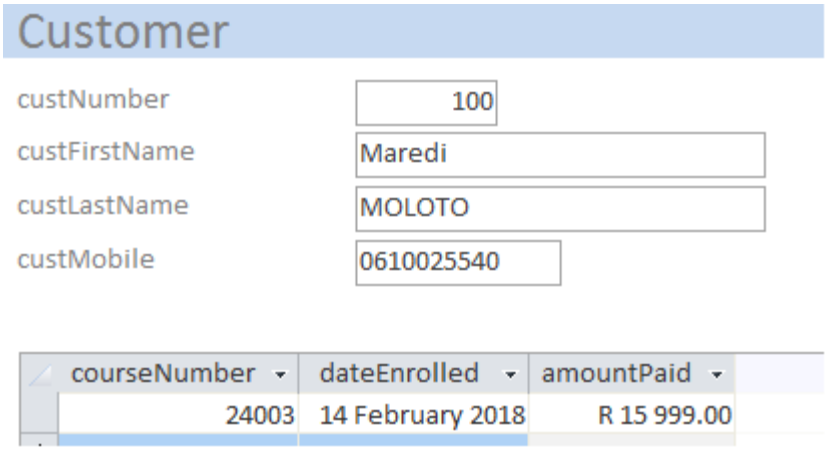


FIGURE 5.1

Make a PRINTOUT of the form.

3.3.2 Use the form created in QUESTION 3.3.1 above to record the course registered by each customer as per TABLE 1 below.

TABLE 1: Enrolment Information

CustNumber	CourseNumber	AmountPaid
100	24002	R2 400,00
100	24004	R1 500,00
101	24003	R1 500,00
102	24002	R1 200,00
104	24002	R500,00
105	24003	R1 200,00

(2 × 1) (2)

3.4 Create a database query using all three tables using the instructions below.

3.4.1 Concatenate the customer name and surname under a column called *Candidate Name*. (1)

3.4.2 Select the following fields:

- Customer Table: custMobile
- Course Table: courseName, courseFee
- Enrolment Table: amountPaid (2)

NOTE: The query must only return payments of more than R1 000,00

3.4.3 Add a column called *Amount Due*, and subtract the *amount paid* from *course fee*. (2)

3.4.4 Save the query as *qryOutstandingFees* and your EXAMINATION NUMBER.

3.4.5 Open the query in design view and make a screen print of the query design. Ensure that ALL columns are clearly visible. (1)

3.4.6 Run the query and print the resulting records. Ensure that ALL columns are clearly visible. (1)

3.5 Create a database report based on the query created in QUESTION 3.4.

FIGURE 5.2 below is a PRINTOUT of the report.

Course Name	Candidate Name	CellPhone	AmountPaid	Course Fee	Amount Due
C++ Programming					
	Anna-Merie VAN LINDEN	0842007783	R 1 200.00	R 14 000.00	R 12 800.00
	Shane SMITH	0832349908	R 1 500.00	R 14 000.00	R 12 500.00
			R 2 700.00	R 28 000.00	R 25 300.00
ICT in Education					
	Kingsley KOLOBE	0731123349	R 500.00	R 12 500.00	R 12 000.00
	Meriam HELEN-AIDE	0726560885	R 1 200.00	R 12 500.00	R 11 300.00
	Maredi MOLOTO	0610025540	R 2 400.00	R 12 500.00	R 10 100.00
			R 4 100.00	R 37 500.00	R 33 400.00
Web design					
	Maredi MOLOTO	0610025540	R 1 500.00	R 9 200.00	R 7 700.00
			R 1 500.00	R 9 200.00	R 7 700.00
			R 8 300.00	R 74 700.00	R 66 400.00

FIGURE 5.2

3.5.1 Show ALL the fields. (1)

3.5.2 Group the report details by Course Name. (1)

3.5.3 Calculate the subtotals and grand totals of amount paid, course fee and amount due. (3)

3.5.4 Change the title to The Course List and your EXAMINATION NUMBER.

3.5.5 Save the report as rptCourse and make a PRINTOUT of the report using landscape orientation. Ensure that all columns are visible. (2)

[25]

TOTAL: 100

Copyright reserved