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)

22 November 2024 (X-paper)
09:00–13:00

CANDIDATES ARE ALLOWED 15 MINUTES TO READ THE INSTRUCTIONS BEFORE THE START OF THE EXAMINATION.

This question paper consists of 17 pages.

491Q2N2422

Copyright reserved

DEPARTMENT OF HIGHER EDUCATION AND TRAINING
REPUBLIC OF SOUTH AFRICA

NATIONAL CERTIFICATE (VOCATIONAL)
PRINCIPLES OF COMPUTER PROGRAMMING
(Second paper)
NQF LEVEL 3
TIME: 4 HOURS
MARKS: 100

INSTRUCTIONS AND INFORMATION

1. Candidates MUST be seated in the computer laboratory 45 minutes before the start of the examination to ensure that the allocated computer workstation has the appropriate software and can make printouts.

2. Answer all the questions.

3. Read ALL the questions carefully.

4. Number the answers according to the numbering system used in this question paper.

5. This question paper is set with computer programming terms that are specific to the BASIC programming language and must be completed using Visual Studio.NET or VB.NET Community/Express together with MS Word and MS Access.

6. Answer the questions according to the instructions and specifications given in each question. Marks will be awarded according to the set requirements.

7. Answer ONLY what is required in each question. For example, if the question does not ask for data validation, then no marks will be awarded for data validation.

8. Routines such as *search*, *sort*, *selection* and *file handling* must be hard-coded by you. You may NOT make use of the built-in features of the programming language.

9. All data structures must be defined by you, unless the question has partially completed source code. If a question has partially completed source code, do NOT delete the source code.

10. Candidates are ALLOWED to make use of the help function of the Integrated Development Environment.

11. All work must be regularly saved on the hard disk/flash disk/network drive that has been allocated, to prevent loss of work. No additional time may be allowed for work lost due to failure to save.

Copyright reserved

12. In the event of problems such as power failure, computer or printing device breakdown, the invigilator will make the necessary arrangements to continue with the examination session. Candidates will not be penalised with time lost under such circumstances.

13. No candidate may print their work or make a flash drive available for another candidate or access another candidate's work via the network. Any attempt to access any information from or transfer information to another candidate in whatever form is a contravention of the examination rules and regulations and will be viewed in an extremely serious light.

14. Candidates must make sure that they print their work immediately after they complete a question. The candidate's EXAMINATION NUMBER must appear on ALL printouts. No answers without a printed EXAMINATION NUMBER will be marked. Source code printouts and screenshot printouts with handwritten examination numbers will NOT be accepted or marked.

15. The source code must be printed directly from the Integrated Development Environment (IDE). Candidates are NOT ALLOWED to copy the source code from the IDE and paste in MS Word for printing purposes.

16. All PRINTOUTS of the source code and screenshots printed during the examination session MUST be handed in to the invigilator.

17. Steps to print a User Interface or any other screen shot:

    17.1 Run/Debug the solution.

    17.2 Hold down the ALT key and press the print screen (Prnt Scrn) key.

    17.3 Open a blank MS Word document and paste the screen shot.

    17.4 Insert a blank THREE columns header.

        (a) Add your EXAMINATION NUMBER against the left margin.
        (b) Add your QUESTION NUMBER against the right margin.

    17.5 Make a PRINTOUT of the page.

18. The following VB.NET solution files will be saved on the computer in the PCP L3 P2 *Examination* folder:

    18.1 QUESTION 1 → …\PCP L3 P2\Question_1\Question_1.sln

    18.2 QUESTION 2 → …\PCP L3 P2\Question_2\Question_2.sln

    18.2 QUESTION 3 → …\PCP L3 P2\Question_3\Question_3.sln

    18.4 QUESTION 4 → …\PCP L3 P2\Question_4\Question_4.accdb

Copyright reserved

QUESTION 1: GENERAL PROGRAMMING

The following questions refer to the VB.NET solution called *Question_1* that has been saved on the workstation.

FIGURE 1 below is a screenprint of the graphical user interface for *Question_1*.

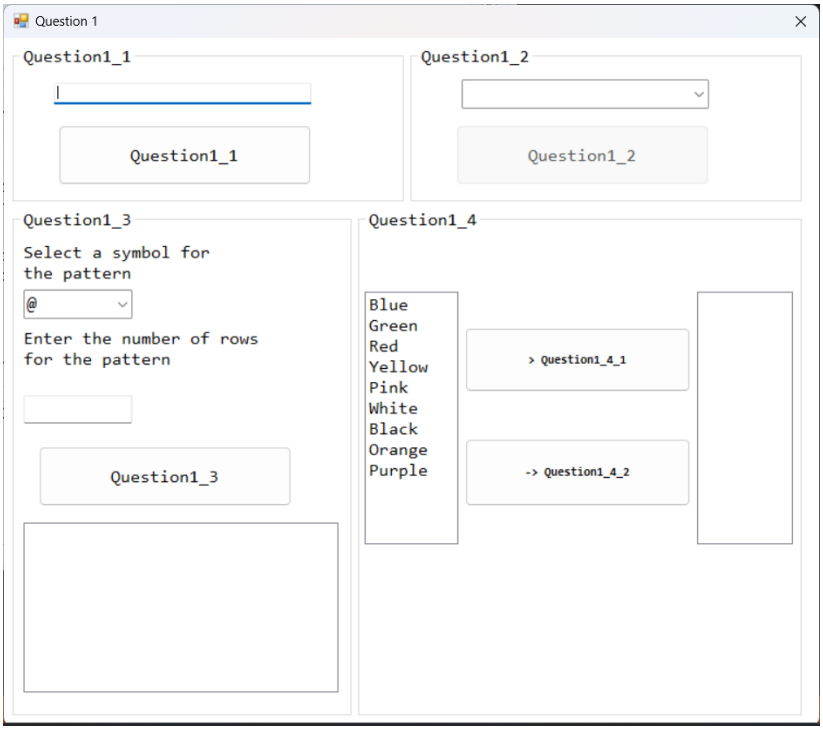


FIGURE 1: SCREENSHOT OF THE GRAPHICAL USER INTERFACE

The user interface displays FOUR (4) different sections named QUESTION1_1 to QUESTION1_4. The completed graphical user interface (GUI), with partially completed source code has been saved on the workstation.

Open the VB.NET solution *Question_1.sln* and follow the instructions below to complete the application.

1.1 When the **Question1_1** button is clicked:

Write code to modify the contents and appearance of the **Question1_1** text box control as follows:

- Change the font size of the text box to 12 pt.
- The text “Hello Mzansi” must appear in the text box.
- Make the text box read-only so that users cannot change its contents.
- Change the background colour of the text box to yellow. (4)

Example of output:



1.2 When the combo box **cboQuestion1_2** *drops down* event occurs:

1.2.1 Write code to do the following:

- Remove all the items in the combo box.
- Populate the combo box control with the names of the twelve months of the year. (3)

Example of output



1.2.2 Enable the **Question1_2** button control. (1)

1.2.3 The user must then select a month from the combo box control **cboQuestion1_2** and click the **Question1_2** button.

When the **Question1_2** button is clicked, write code to do the following:

- Determine if the combo box control has been selected.
- Remove the month that has just been selected from the combo box control. (4)

1.3 When the **Question1_3** button is clicked:

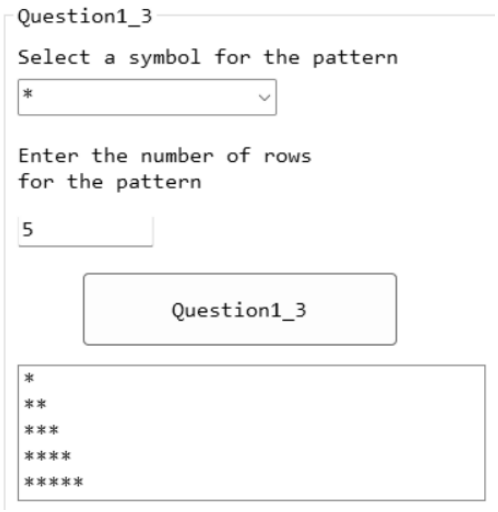
A pattern with a symbol selected from the combo box must be displayed. **Note**: Seven different symbols have already been populated into the combo box.

The pattern is a triangle that consists of the number of rows that is obtained from the text box.

Write code to do the following:

- Clear the list box.
- Assign the symbol that has been selected by the user from the combo box to a local string variable.
- Assign the number of rows from the text box to an integer variable.
- Use a loop structure to draw the pattern using the selected symbol in the list box control. (7)

Example of output:



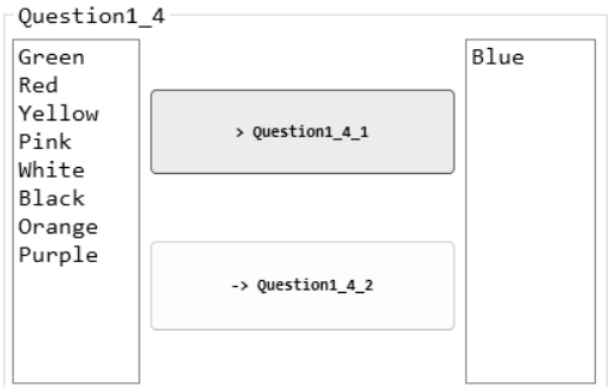
1.4 Write VB.NET code to implement a functionality that allows you to move items from one ListBox (source) to another ListBox (destination).

1.4.1 When the **Question1_4_1** button is clicked, move ONLY the selected item from the source list box to the destination list box.

Write code to do the following:

- Check if any item has been selected in the source list box.
- Add the selected item from the source list box to the destination list box.
- Remove the selected item from the source list box ONLY. (3)

Example of output if the colour *Blue* is selected and the *Question1_4_1* button is clicked:

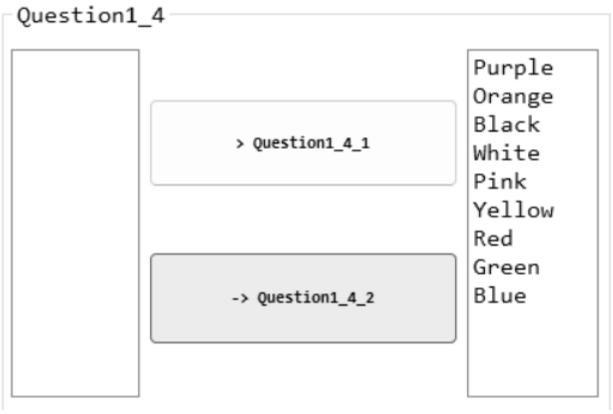


1.4.2 When the **Question1_4_2** button is clicked, move ALL the items from the source list box to the destination list box.

Write code to do the following:

- Loop through all the items in the source list box.
- Whilst looping:
  - Move the item(s) from the source list box to the destination list box.
  - Remove the item(s). (3)

Example of output if the *Question1_4_2* button is clicked:



1.5

- Enter your examination number, computer workstation number and date as comments in the first three (3) lines of the source code.
- Save your source code.
- Make a printout of the source code. (1)

[26]

QUESTION 2: MODULAR PROGRAMMING

The following questions refer to the VB.NET solution called *'Question_2'* that has been saved on the workstation.

Mzansi Movers is a well-established logistics and moving company operating in South Africa. They specialise in moving boxes of varying weights. Mzansi Movers offers a variety of light delivery vehicles (vans) with different load capacities. These load capacities range from 500 kg to 1500 kg. This selection of vans with varying load capacities allows customers to choose the most suitable option based on their specific moving or transportation needs.

They have recently started a project to develop a user-friendly app that allows customers to calculate transportation costs easily. The application allows users to select a specific van based on weight they estimate they will need. Users will be asked to input the distance in kilometres to their destination. Users will then input the weight of each individual box. The application will then give a quotation on the cost of moving the goods.

Copyright reserved

Mzansi movers calculates the cost based on a fixed cost as set out below in TABLE 1.

Weight charge	Transport charge
R2,80 per kilogram	R1.10 per kilometre

TABLE 1: Charges

They require assistance in completing the VB.Net application. ☆

FIGURE 2 and FIGURE 3 below are screenprints of the graphical user interface. The completed interface has been saved on the workstation.



FIGURE 2

☆

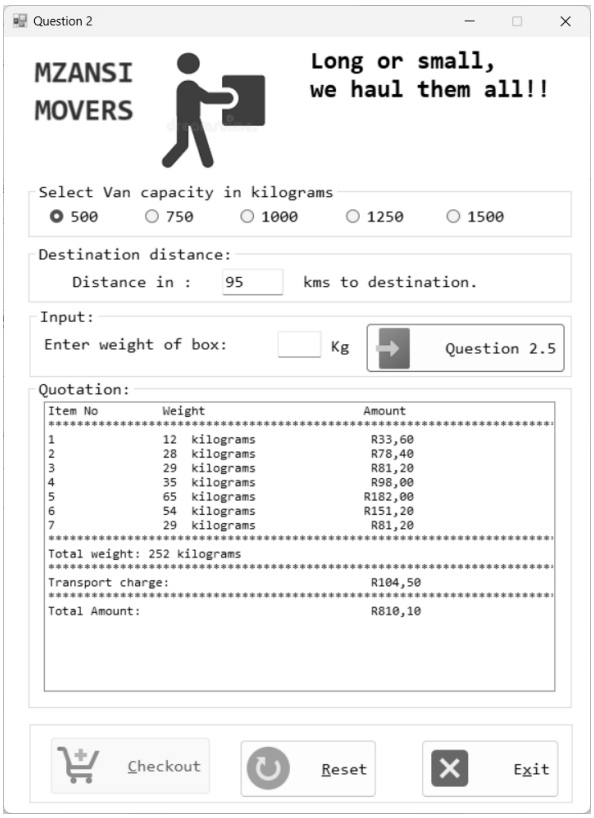


FIGURE 3

Boxes can be added as long as the sum of the current load and the weight of the box falls within the van's capacity. The van doesn't need to be completely full of boxes before it leaves, and the input process can be stopped before reaching the full capacity of the van.

It's also possible that a single parcel might be too heavy, causing the van's capacity to be exceeded. In such cases, a replacement parcel with an acceptable weight can be substituted, as shown in FIGURE 4 below.

Copyright reserved

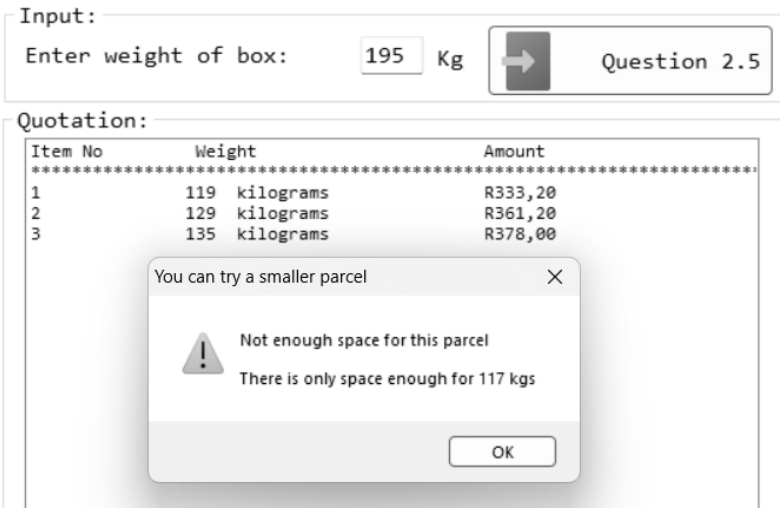


FIGURE 4

The user then has a choice of adding a smaller box or not, as shown in FIGURE 5 below.

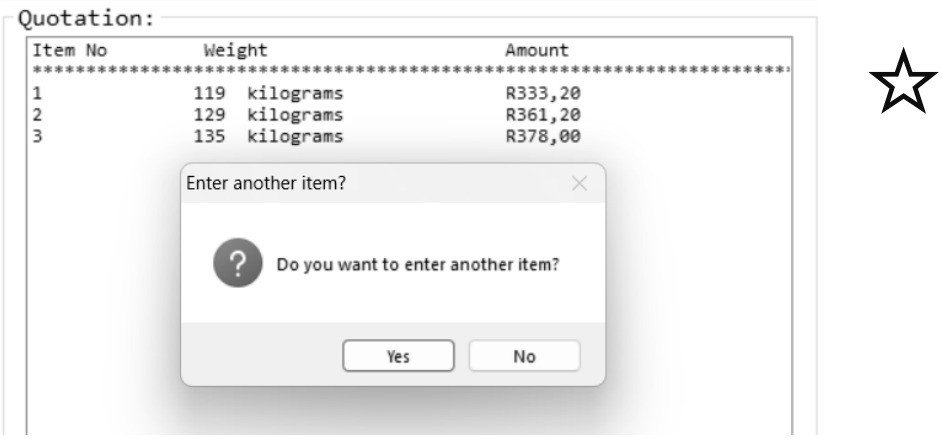


FIGURE 5

If the user selects NO, then the application calculates the total cost and displays the total in the list box control.

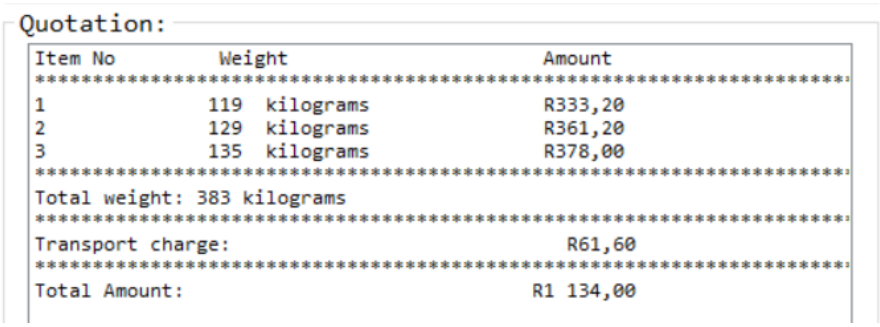


FIGURE 6

Copyright reserved

The IPO diagram for the application is shown in TABLE 2 below.

INPUT	PROCESS	OUTPUT
Van capacity Destination distance Weight of boxes		Total Weight Transport Charge Total Amount

TABLE 2: IPO DIAGRAM

The user interface displays FOUR (4) sections, three for the data input and the other to provide a quotation. The VB.Net application has been partially coded.

Open the VB.NET solution Question_2**.**sln and follow the instructions below to complete the application.

2.1 Declare the following at Class level:

2.1.1 TWO *constants* which will be assigned the weight and transport charges as shown in TABLE 1. (1)

2.1.2 THREE *single dimension*, *dynamic* arrays which will be assigned the values for the *Item Number*, *Weight* and *Amount*. (1)

2.1.3 An *integer* variable used for counting. (1)

2.2 Write code to create a user-defined function procedure called *DetermineCapacity*. The procedure must determine which of the capacity radio buttons have been selected and return the capacity as an *integer* value back to the calling event procedure. (5)

2.3 Write code to create a user-defined function procedure called *CalcWeightCost*, which will receive a copy of the *Weight* as a double-precision floating-point number parameter. The procedure must calculate and return the *Rand Value* of the *Weight Cost* based on the following formula:

*Rand Value = Weight * Weight Charge* (5)

HINT: *Weight Charge* was declared as a constant in QUESTION 2.1.1.

2.4 Write code to create a user-defined function procedure called *CalcDistanceCost*, which will receive a copy of the *Distance* as a whole number parameter. The procedure must calculate and return the *Rand Value* of the *Distance Cost* based on the following formula:

*Rand Value = Distance * Transport Charge* (5)

HINT: *Transport Charge* was declared as a constant in QUESTION 2.1.1.

2.5 When the **Question2.5** button is clicked:

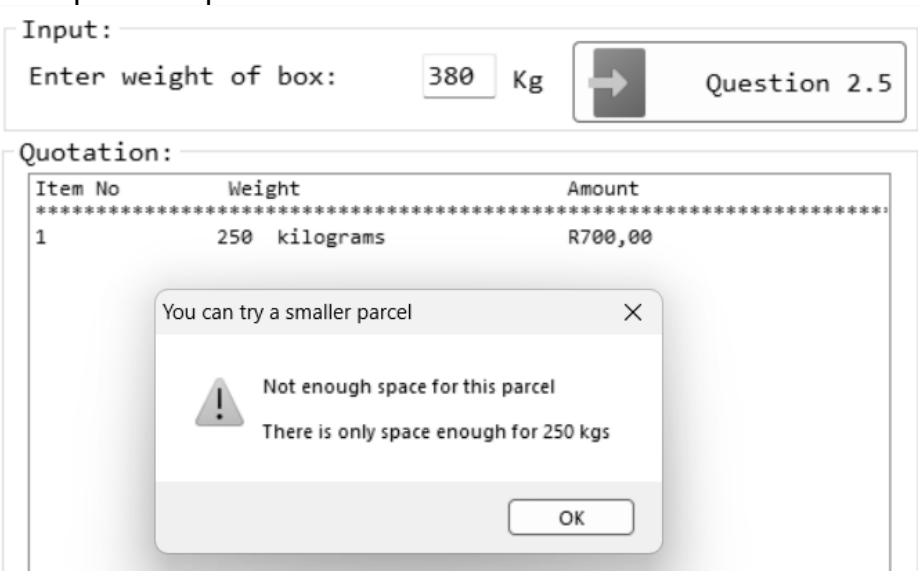
2.5.1 Write code to *resize* AND *preserve* the existing elements of the THREE arrays created in QUESTION 2.1.2 according to the *integer* counter variable created in QUESTION 2.1.3. (3)

2.5.2 Write code to determine if the *current capacity* of the van is greater than the *maximum capacity* of the van.

If TRUE.

- Create a pop-up dialog box to display the error to the user (see example of output below).
- Clear the contents of the weight text box.
- Call the sub-procedure *ConfirmAdd()*.
- Terminate the procedure. (5)

Example of output:



2.5.3 Write code to populate the arrays for the *Item Number* and *Weight*. (2)

2.5.4 Write code to populate the array for the *Amount* using the function created in QUESTION 2.3. (2)

2.6

- Enter your examination number, computer workstation number and date as comments in the first three (3) lines of the source code. (1)
- Save your source code.
- Make a printout of the source code.

[31]

QUESTION 3: FILE AND ERROR HANDLING

The following questions refer to the VB.NET solution called *'Question_3'* that has been saved on the workstation.

The Deputy Principal: Registrations at Mzansi TVET College has requested your assistance in designing an application to help with student induction. Mzansi TVET College has FIVE venues where student induction will take place. The application captures student information, such as their name, their course of study and a venue where the induction for the student will take place. The application then saves the information to a delimited text (.txt) file.

FIGURE 7 below is a screenprint of what the graphical user interface looks like. The completed interface has been saved on the workstation.

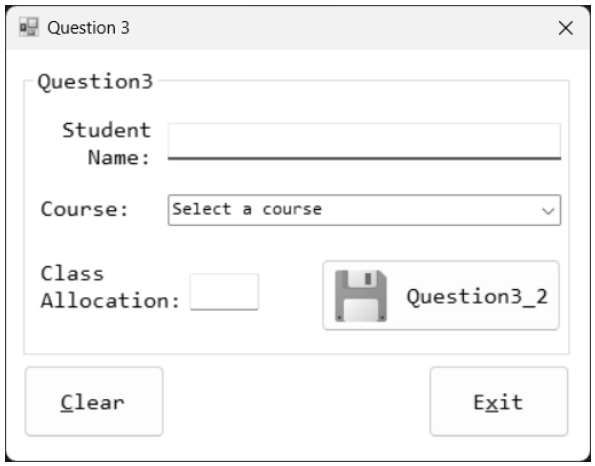


FIGURE 7: SCREENSHOT OF INTERFACE

The user interface displays ONE (1) section, for the data input of relevant student information. After entering the information, the data will be saved to a delimited text file called *Students.txt*.

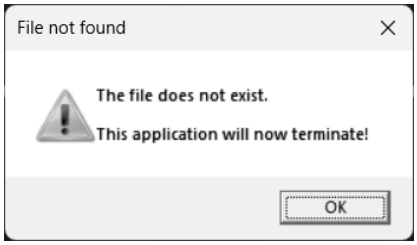
Open the VB.NET solution Question_3**.**sln and follow the instructions below to complete the application.

3.1 Write code to the user-defined sub-procedure called *CheckFile()*. The procedure must do the following:

- Use exception handling code to check whether the text file *Students.txt* exists.

If the text file does not exist, the application must display an appropriate message and exit the application. (3)

Example of output if the file does not exist:



☆

3.2 When the **Question3.2** button is clicked:

Write appropriate *error handling* code that will make use of a *StreamWriter* object to save the information entered in the text boxes to the delimited text file *Students.txt*. (7)

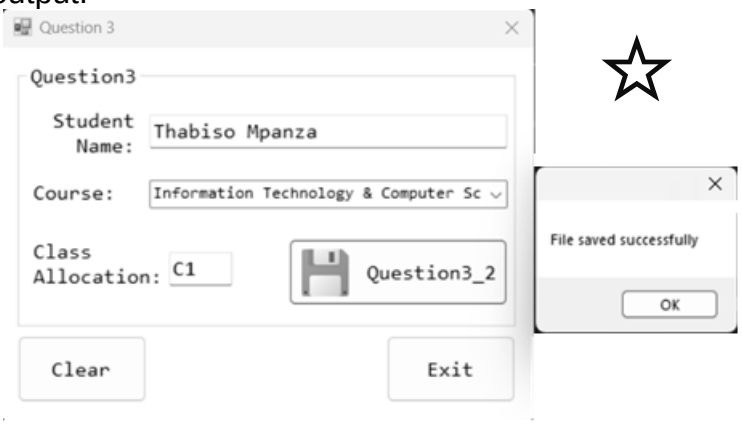
The format of each line of text in the text file is as follows:

<Student Name>,<Course>#<Class>

Example of the expected first line of text in the text file:
Thabiso Mpanza, Information Technology & Computer Science#C1

Display an appropriate message to inform the user that the file has been saved and clear all the text boxes.

Example of output:



☆

3.3
- Enter your examination number, computer workstation number and date as comments in the first three (3) lines of the source code. (1)
- Save your source code.
- Make a printout of the source code.

[11]

QUESTION 4: DATABASE

The Senior Lecturer at Mzansi TVET college has asked you to assist in creating a database to assist with the allocation of student enrolments. The IT lecturer has already created a skeleton database called *Question_4.accdb*. The database already contains TWO tables (STUDENT and CLASS).

The IT lecturer also said that in database modelling the relationship between "STUDENT" and "CLASS," is a "Many-to-Many" relationship. This is because many students can be enrolled in many classes, and each class can have multiple students. A many-to-many relationship can be considered a less desirable database design due to its complexity, potential performance issues, and challenges in data integrity maintenance.

Below is the design specification of the two tables.

	FIELD NAME	DATA TYPE	FIELD SIZE	DESCRIPTION
🔑	Stu_ID	Short Text	13	Unique code.
	Stu_Name	Short Text	25	First Name(s) of student
	Stu_Surname	Short Text	30	Surname of student

TABLE 3: STUDENT

	FIELD NAME	DATA TYPE	FIELD SIZE	DESCRIPTION
🔑	Class_ID	Autonumber	Integer	Automatic unique code.
	Class_Name	Short Text	25	Name of the class
	Class_Description	Short Text	125	Description

TABLE 4: CLASS

NOTE: 🔑 denotes Primary Key (PK)

To implement a Many-to-Many relationship in a relational database, you typically use an additional junction table or linking table that contains foreign keys to both the "STUDENT" and "CLASS" tables. This junction table allows you to associate which students are enrolled in which classes and vice versa. They help eliminate data redundancy by storing only the necessary relationships without duplicating data.

The IT lecturer drew a relationship diagram (FIGURE 8) to show you what the final design should look like.

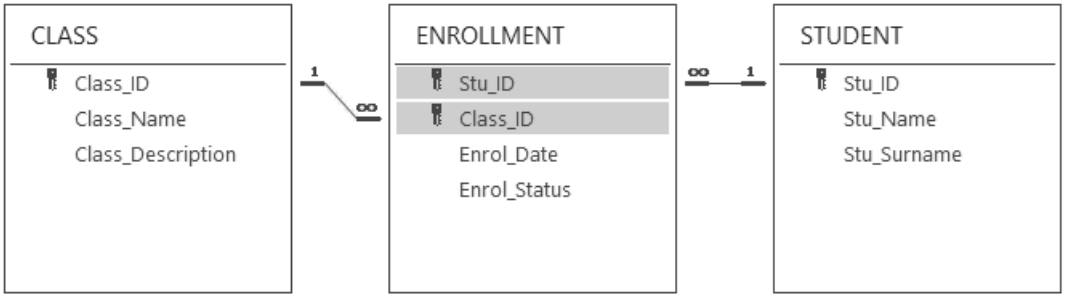


FIGURE 8: DATABASE DESIGN RELATIONSHIPS

4.1 Rename the database to include your EXAMINATION NUMBER. For example, Question_4.accdb - 020427XXXX089. (0)

4.2 Open the database and follow the instructions below.

4.2.1 Create the Junction table *ENROLLMENT* using the information provided in TABLE 5 below.

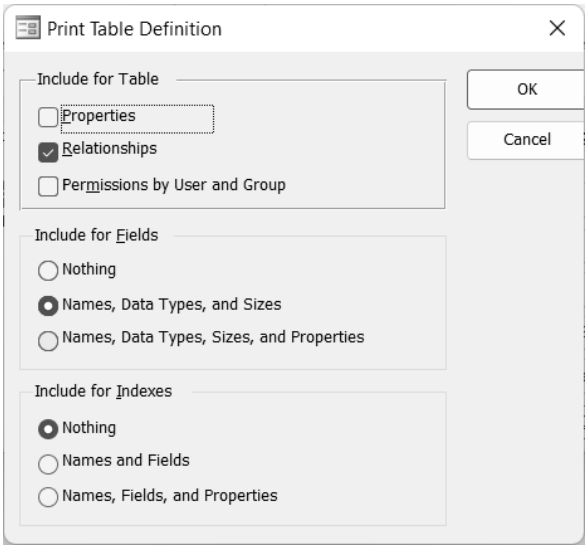
☆

	FIELD NAME	DATA TYPE	FIELD SIZE	TYPE
🔑	Stu_ID			Composite Primary Key/ Foreign Key
🔑	Class_ID			
	Enroll_Date	Date/Time		
	Enroll_Status	Short Text	25	

TABLE 5: ENROLLMENT

4.2.2 Create the appropriate relationship between all THREE tables. Enforce referential integrity. Refer to FIGURE 8 above.

4.2.3 Use the database documenter to set the print table definition options as shown in FIGURE 9 below.



☆

FIGURE 9: TABLE STRUCTURE SETTINGS

Print the database table structure for the *Enrollment* table ONLY. (9)

Copyright reserved

4.3 Open the ENROLLMENT table. Capture the following enrollments.

STUDENT ID	CLASS ID	DATE ENROLLED	ENROLLED STATUS
0001105010089	1	01 Nov 2023	Enrolled
0002250145086	3	02 Nov 2023	Waiting List
0012251235081	3	02 Nov 2023	Waiting List
0012255221087	6	18 Oct 2023	Enrolled
0102141245084	2	06 Nov 2023	Enrolled
0102141245084	5	21 Nov 2023	Waiting List
0111253215081	4	13 Nov 2023	Waiting List

Save ALL the records.

4.3.1 Print ALL the records (4)

4.4 Create a *database query* in design view using ALL tables.

The query must retrieve the *Stu_ID*, *Stu_Name, Stu_Surname* and *Class_Name* sorted by Surname in ascending order, for ALL students that have enrolled in a class. The query must ONLY display and format the field names as shown below.

Example of output:

Name	Surname	Class
Harvey	Cobblepot	Database Systems
Dick	Grayson	Algorithms
David	Lee	Database Systems
Peter	Pakade	Introduction to CS
Peter	Pakade	Web Development
Patsy	Walker	Artificial Intelligence
Bruce	Wayne	Data Structures

Save the query as Q 4.4 with your EXAMINATION NUMBER.

4.4.1 Open the *Query* in Design View, ensure that all columns are clearly visible and make a screen print of the query in design view. (6)

4.5 Create a *database query* in design view using ALL tables.

The query must retrieve the *Stu_ID*, *Stu_Name, Stu_Surname*, *Class_Name* and *Enrol_Status* sorted by Class in descending order for ALL students that are on the *Waiting List*. The query must ONLY display and format the field names as shown below.

Example of output:

Student ID	Name	Surname	Class	Status
0102141245084	Peter	Pakade	Web Development	Waiting List
0012251235081	Harvey	Cobblepot	Database Systems	Waiting List
0002250145086	David	Lee	Database Systems	Waiting List
0111253215081	Patsy	Walker	Artificial Intelligence	Waiting List

☆

Save the query as Q 4.5 with your EXAMINATION NUMBER.

4.5.1 Open the *Query* in Design View, ensure that all columns are clearly visible and make a screen print of the query in design view. (6)

4.6 Create the following *database report* using the query created in QUESTION 4.5 above.

ENROLLMENT - Examination number

Class	Student ID	Name	Surname	Status
Artificial Intelligence				
	0111253215081	Patsy	Walker	Waiting List
Database Systems				
	0012251235081	Harvey	Cobblepot	Waiting List
	0002250145086	David	Lee	Waiting List
Web Development				
	0102141245084	Peter	Pakade	Waiting List

03 November 2023 Page 1 of 1

☆

Group the report details according to the class name.

Save the *report* as *Waiting_List* with your examination number. Ensure that your examination number appears in the *report header*.

4.6.1 Print the report. (7)

[32]

TOTAL: 100

Copyright reserved