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)

11 December 2020 (Y-paper)
13:00–17:00

This question paper consists of 18 pages.

575Q2N2011

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.

4. Be seated in the computer laboratory 30 minutes before the start of the examination to ensure that the allocated computer has the appropriate software and can make printouts.

5. This question paper is set with programming terms specific to the BASIC programming language and must be completed using Visual Studio.NET or VB.NET Express together with MS Office or applicable word-processing software.

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.

8. You must hard code routines such as *search*, *sort*, *selection* and *file handling* and you may not use the built-in features of the programming language.

9. Define all data structures unless the question has partially completed source code.

10. Do not delete the source code if a question has partially completed source code.

11. You may use the *Help* function of the IDE.

12. Save your work regularly on the hard disk/flash disk/network drive that you have been given to prevent loss of work. No additional time may be allowed for work lost due to lack of saving.

13. In the event of problems such as power failure or computer/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.

Copyright reserved

14. You may not print your 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 manner.

15. Print your work immediately after completing a question. Your EXAMINATION NUMBER must appear on all printouts. No questions without a printed EXAMINATION NUMBER will be marked. Source code printouts and screenshot printouts with handwritten examination numbers will not be accepted or marked.

16. Print the source code directly from the integrated development environment (IDE). You are not allowed to copy the source code from the IDE and paste in MS Word for printing purposes.

17. Hand all printouts of the source code and screenshots printed during the examination session to the invigilator.

18. Steps to print a user interface or any other screenshot:

    18.1 Run/Debug the solution (for Visual Studio.NET only).

    18.2 Hold down the ALT key and press the *Print Screen* key.

    18.3 Open a blank MS Word document and paste the screenshot.

    18.4 Insert a blank three-column header.

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

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

    18.5 Make a printout of the page.

19. The following VB.NET solution files will be saved on the computer in the Principles of Computer Programming L3 P2 *Examination* folder:

    19.1 QUESTION 1
    …\PCP L3 P2\...\Wordalizer.sln

    19.2 QUESTION 2
    …\PCP L3 P2\...\ClassReps.sln & …\PCP L3 P2\...\Debug\ClassList.txt

    19.3 QUESTION 3
    … No files

20. Write neatly and legibly.

Copyright reserved

QUESTION 1: FUNCTIONS, SUBPROCEDURES AND FILE HANDLING

The following questions refer to the VB.NET solution named *Wordalizer* that has been saved on the computer.

The NCV English Level 2 lecturer at your college asked you to design a VB.NET application to assist students in analysing some English words. The application must take as input a string of letters and analyse it to determine how many letters are vowels and how many letters are consonants. The lecturer has provided the following information to help you get started.

Vowels: The letters A, E, I, O, U from the alphabet
Consonants: All other letters of the alphabet except the vowels

FIGURE 1 is a screenshot of the graphical user interface (GUI).

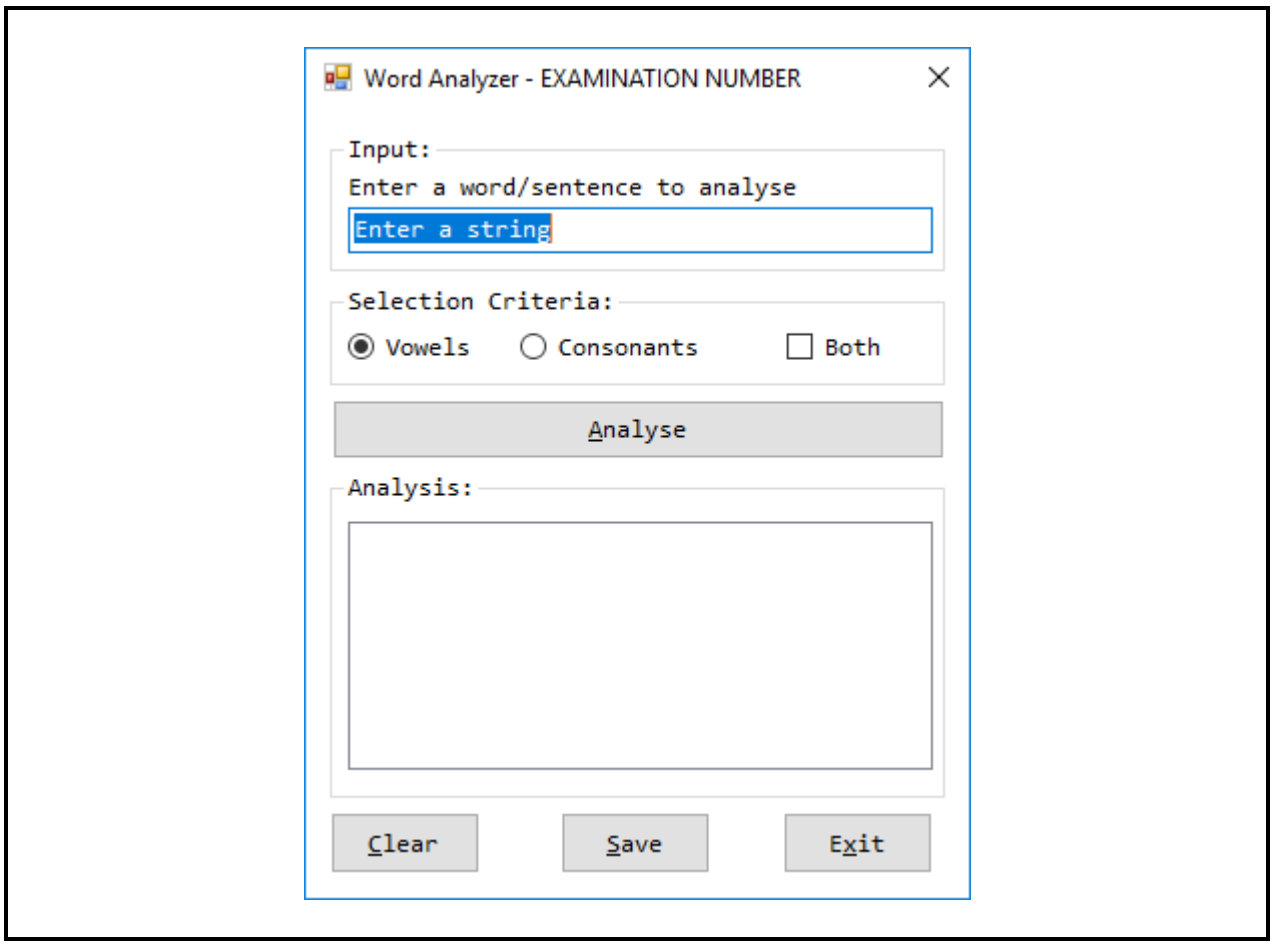


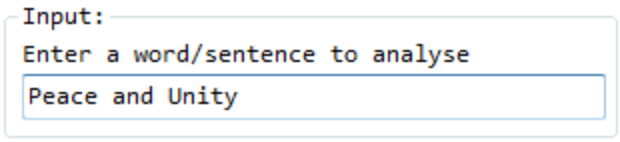
FIGURE 1

Use the IPO chart below to assist you with the computer program development:

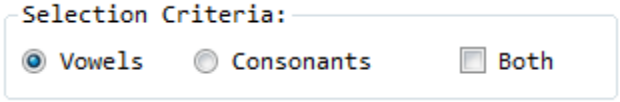
INPUT	PROCESSING	OUTPUT
Word/Sentence	InputString = LowerCase(Word/Sentence) Vowel Count = Case {a,e,i,o,u} Consonant = Word Count – Vowel Count	Word Count Vowel Count Consonant Count Output Text File

The following steps explain how to use the *Wordalizer* application.

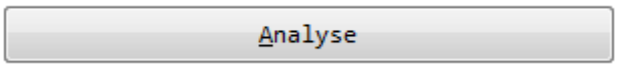
STEP 1 Run the application and enter the word/sentence to analyse.



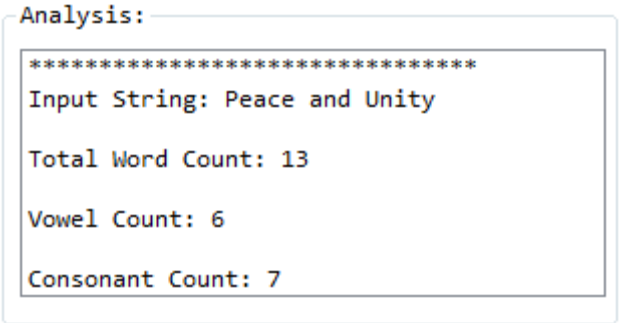
STEP 2 Select either the *Vowels* or *Consonants* radio button or the *Both* checkbox.



STEP 3 Click the *Analyse* button.



The application will display the results in the list-box control.

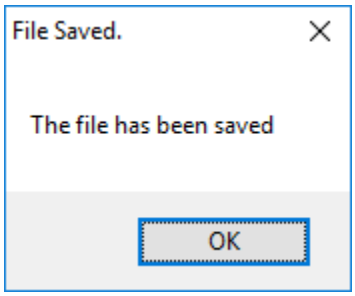


STEP 4 Click on the *Save* button to save the results to a text file.



Copyright reserved

The application will confirm the save with a pop-up dialog box.



The completed graphical user interface (GUI) has been saved on your computer.

Open the VB.NET solution *Wordalizer.sln* and follow the following instructions to complete the application:

1.1 Declare an appropriate form-level variable which will be assigned the value of the *InputString*. (1)

1.2 In the *load* event of the interface *frmWordalizer*:

1.2.1 Remove any text from the *InputString* text box using method only (1)

1.2.2 Enable the *Timer* control (1)

1.3 Create a user-defined function procedure named *CountVowel* which will perform the following:

1.3.1 The function procedure will accept as input ONE parameter named *InputString* by *pointing* to the original variable and will return a *whole* number. (4)

1.3.2 Declare an integer variable named *Count* and initialise to zero. The variable will later be assigned the value of the number of time(s) a vowel appears in the input string. (1)

1.3.3 Create a *For Each* loop which will perform the following:

(a) Iterate through each *letter* as a *character* in the *InputString*. (4)

(b) Create a *Select Case* statement inside the *For Each* loop with the following specifications:

- *testexpression* → *letter*
- *expressionlist* → X 5 for each letter of the vowels

NOTE: Use the character conversion function. (2)

(c) Increment the variable created in QUESTION 1.3.2 by a value of ONE in each of the cases defined in QUESTION 1.3.3 (b).

HINT: The *Select Case* statement must try to match the *InputString* to each one of the vowel letters. When a match is made the variable *Count* is incremented by a value of 1. (2)

Copyright reserved

1.3.4 Return the *counter* variable to the calling procedure. (1)

1.4 Create a user-defined function named *CalcConsonant* which will accept as input TWO parameters, named *VowelCount* and *WordCount,* as whole numbers and as copies. The function procedure must simply return the *difference* between *WordCount* and *VowelCount* as a whole number. (4)

1.5 When the *Both* checkbox is *ticked* the application must determine if both checkboxes have been ticked or not and:

1.5.1 If *ticked* the application must *deselect* both radio buttons and then *disable* both radio buttons. (3)

1.5.2 If not *ticked* the application must *enable* both radio buttons. (1)

HINT: Make use of an *If … ElseIf* control structure.

1.6 Add the following program code to the *click event* of the *Analyse* button:

1.6.1 Check if the value entered in the *InputString* text box is a string. If not:

(a) Alert the user with an appropriate message

(b) Select all the *text* in the text box control

(c) Set focus to the text box control

(d) Abort the procedure

HINT: Use the built-in functions with negative logic. (4)

1.6.2 Assign the text from the *InputString* textbox to the variable created in QUESTION 1.1 by converting the text to *lower case* and by *removing* any spaces in the text.

HINT: Use two built-in functions to change to *lower case* and to *remove* spaces from the *InputString* (2)

1.6.3 Invoke the user-defined subprocedure *Display*. (1)

Copyright reserved

1.7 Add the following program code to the *click event* of the *Save* button:

1.7.1 Add a *Try … Catch* statement that will catch any unhandled exceptions that might occur. The catch block should:

(a) Display the message contained in the exception object using a *pop-up* dialog box with an *OK* button and a *Stop* icon

(b) Invoke the *ResetInterface* procedure

(c) Abort the procedure (5)

The questions that follow must be included inside the *Try … Catch* statement created in QUESTION 1.7.1.

1.7.2 Use the *StreamWriter* class to create a *new StreamWriter* object named *SaveToFile* which will save the contents of the list box to a text file named *Results.txt*.

NOTE: Do not include a file path for saving the text file. (3)

1.7.3 Create a *For* loop that will iterate through all the *items* in the *Analysis* list box and then use the *StreamWriter* object created in QUESTION 1.7.1 to *write* each *line* to the text file. (2)

1.7.4 Create a pop-up dialog box as shown in FIGURE 2 to confirm that the file has been saved.

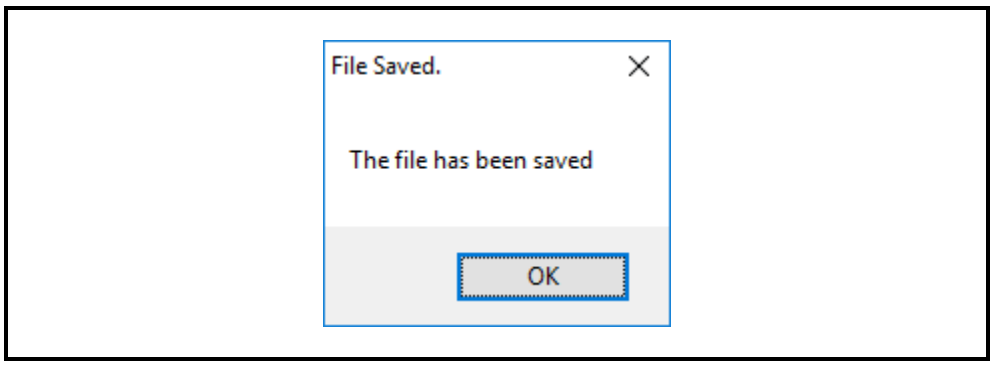


FIGURE 2 (1)

1.8 In the *click event* of the *Clear* button invoke the user-defined subprocedure *ResetInterface*. (1)

1.9 Make a printout of the source code.

1.10 Change the text in the *title bar* of the interface to your EXAMINATION NUMBER. Debug/Run the application, enter the following test data and make a screenshot printout of the interface:

Input String: peace and unity
Criteria: Both (1)

[45]

QUESTION 2: ARRAYS AND FILE HANDLING

The following questions refer to the VB.NET solution named *ClassRep* that has been saved on the computer.

A TVET college lecturer had her class list neatly typed using the Notepad application. The lecturer wants you to develop a class-representative voting system to select a class representative and deputy class representative. The application should read the names of the candidates from a text file, allow students to choose candidates and vote for their representatives.

A maximum of four candidates may be nominated for the two positions in a classroom of 15 students. All students enrolled for the subject are eligible to participate as candidates or to vote for preferred candidates.

The candidate who has the most votes will be the class representative and the deputy class representative position will be given to the candidate with the second most votes. The voting process starts with the nomination of candidates, the voting process and counting of votes.

FIGURE 3 and FIGURE 4 are screenshots of the graphical user interface:

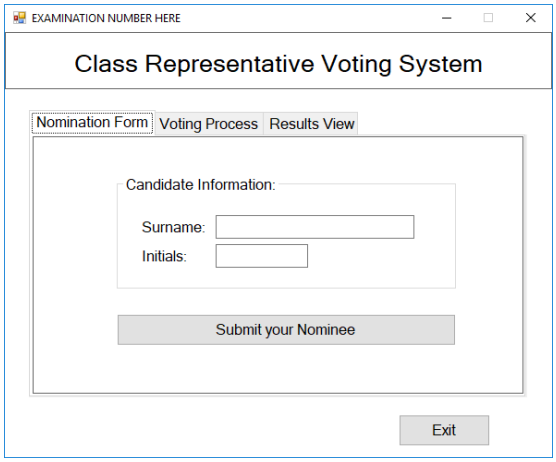


FIGURE 3

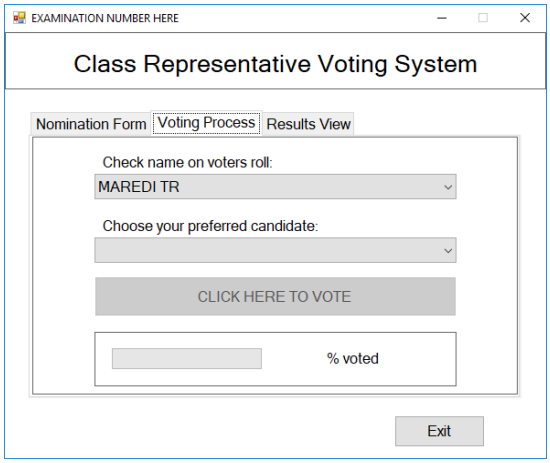


FIGURE 4

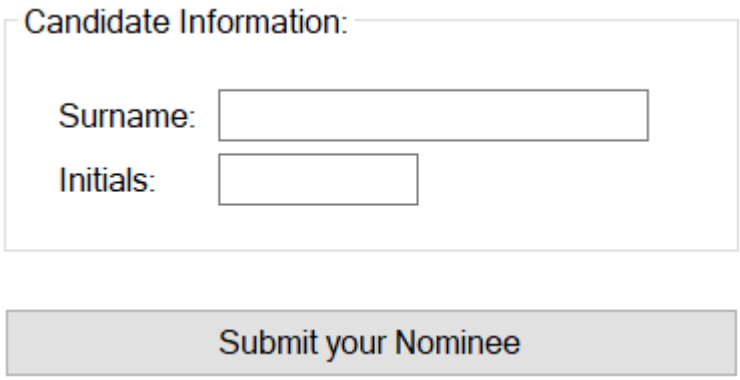
Use the IPO chart below to help you with the computer program development.

INPUT	PROCESSING	OUTPUT
Number of candidates	Read students' names from text file into an array.	All votes cast by students
Maximum number of students in classroom	Populate a combo box to be used as the voters roll with students' names. Enter the names of candidates into an array.	Total votes per candidate
Names of candidates	Populate a combo box with the names of all candidates.	
Names of classroom members	Assign the votes of students into an array.	
	Remove student names from the combo box immediately after casting a vote.	
	Determine the total votes for each candidate.	
	Display the voting results.	

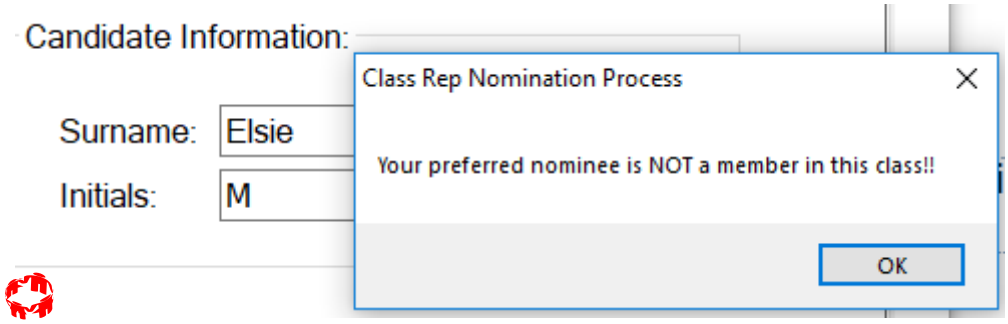
The following steps explain how to use the program.

STEP 1 Run the application.

Enter the preferred candidate surname and initials and click the *Submit* button.

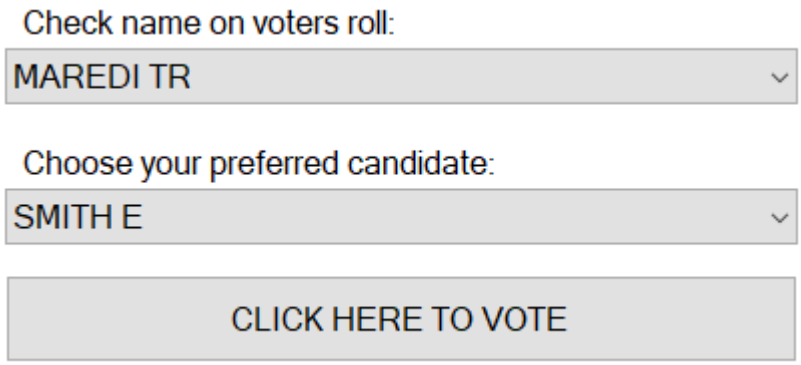


If the candidate name does not exist in the class register, the application should alert the user with an appropriate message as shown below.

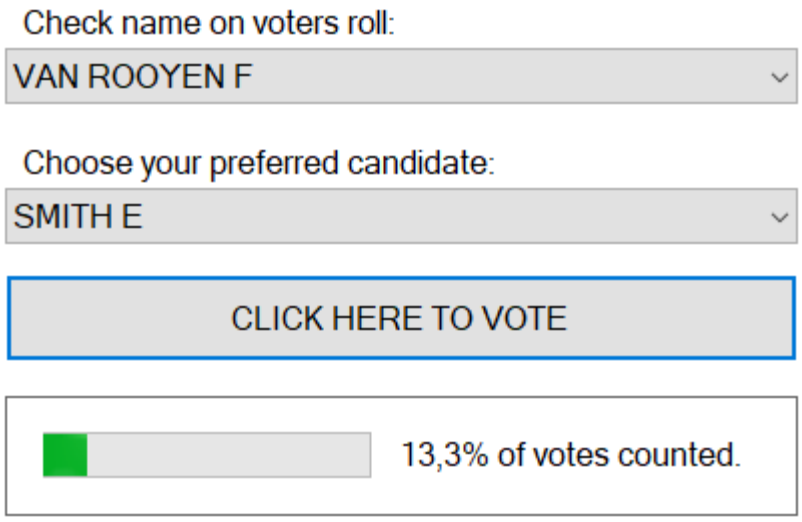


Copyright reserved

STEP 2 Check the name of the student voter if it appears in the combo box. If TRUE select the candidate and click on the *Click Here To Vote* button once.



The application will remove the name of student who has voted.

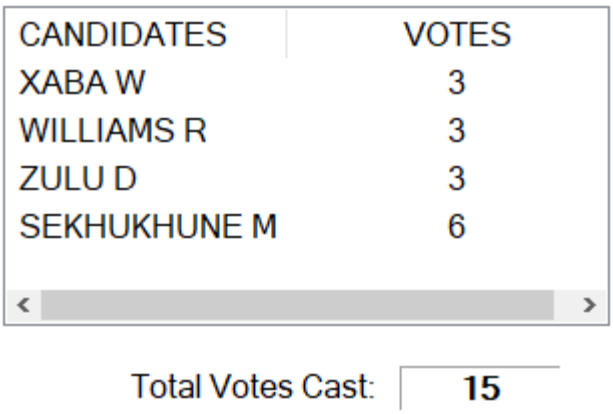


STEP 3 When all students have voted, click on the *Results View* tab and click the *Click To View* button to display the voting outcome.



Copyright reserved

When the *Click To View* button has been clicked the application will produce the output as illustrated below.



The completed user interface with partially completed source code has been saved on the computer. Open the VB.NET solution named *ClassRep.sln* and follow the following instructions to complete the application.

2.1 Declare at form level the following data structures to be used as storage for the input data and processed data (i.e. output):

HINT: Refer to the IPO diagram.

2.1.1 TWO constants – ONE for the maximum number of candidates and ONE for the maximum number of students in the class (1)

2.1.2 A *fixed* single-dimensional array for the number candidates (1)

2.1.3 TWO *dynamic* single-dimensional arrays for the class register and votes (1)

2.2 Add the following code to the user-defined subprocedure named *GenerateRegister*:

2.2.1 Resize the *ClassRegister* array without losing any data that is in the array. (1)

2.2.2 Add the required code to the *StreamReader* object named *ReadFile* to read the data from the text file and assign it to the *ClassRegister* array.

NOTE: The text file has already been included in the *Debug* folder. (1)

2.3 In the form *load* event subprocedure add the required code to:

2.3.1 Invoke the subprocedure named *GenerateRegister* (1)

2.3.2 Populate the *VotersList* combo box with the student names from a single-dimensional array.

HINT: Use a *For … Next* loop to iterate through the array. (4)

Copyright reserved

2.4 Create a user-defined function procedure named *EligibleToStand* with ONE parameter named *Name* to determine if the candidate's name appears in the class register. Complete the function by following the following instructions: (1)

2.4.1 Declare the following TWO variables:

- A flag variable named *status* set to FALSE.
- A loop counter integer variable named i set to 0. (1)

2.4.2 Use a *Do Until … Loop* to iterate throughout the *class register* array and *if* the candidate's name is found, set the flag to TRUE and terminate the loop.

HINT: The loop condition will also require a loop counter variable which must be incremented. (5)

2.4.3 Return the flag variable. (1)

2.5 In the click event of *Submit Nominee* button:

2.5.1 Assign the nominee's name into an appropriate array, increment the counter and assure the user with a message as shown in FIGURE 5.

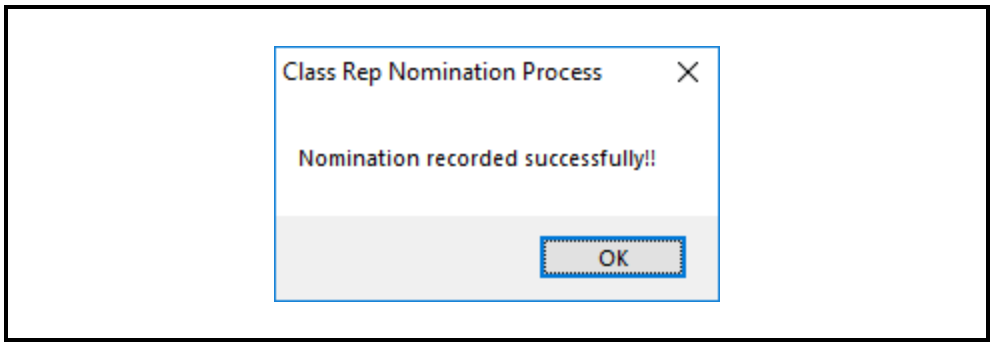


FIGURE 5 (3)

2.5.2 Populate the *Candidates* combo box with the nominee names from a single-dimensional array.

HINT: Use *For … Next* to iterate through the array. (4)

2.6 Add the following program code to the click event of the *Cast Vote* button:

2.6.1 Assign the *selected index* of the candidate chosen from the combo box into an appropriate single-dimensional array. (2)

2.6.2 Increment the variable keeping track of number of votes casted. (1)

2.6.3 Remove the student who had voted from the combo box. (2)

2.7 Add your EXAMINATION NUMBER in the source code. Make a printout of the source code.

[30]

QUESTION 3

The head of the NCV IT department at your college asked you to design a database using MS Access. This database application will be used by lecturers to view individual assessment marks of students.

There are three types of assessments, namely Practical Assessment Task, Formal Test and Internal Examination.

FIGURE 6 is a graphical representation of the database tables and relationships that exist between the various tables.

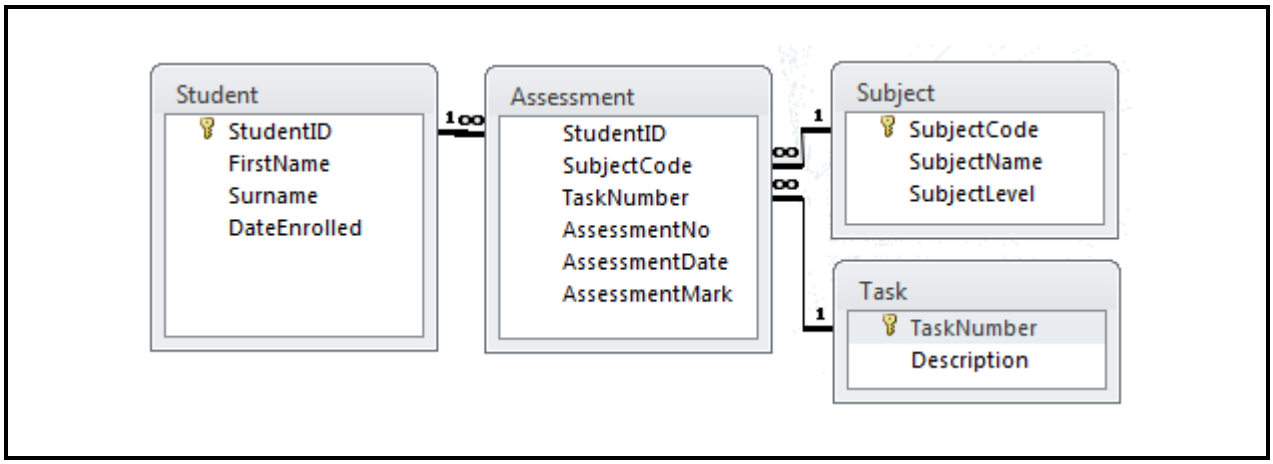


FIGURE 6

Use the information below to set up the database tables in MS Access. Open a new blank database and save it as *NCVStudents* and your EXAMINATION NUMBER.

Create the student table and save it as *Student*.

Field name	Data type	Field size	Format	Description
StudentNumber	Text	13		Primary Key
FirstName	Text	30		
Surname	Text	30		
DateEnrolled	Date		Short date	

TABLE 1: STUDENT

Create the subject table and save it as *Subjects*.

Field name	Data type	Field size	Description
SubjectCode	Text	6	Primary Key
SubjectName	Text	25	
SubjectLevel	Text	2	

TABLE 2: SUBJECTS

Create the assessment tasks table and save it as *Task*.

Field name	Data type	Field size	Description
TaskNumber	Number	Byte	Primary Key
Description	Text	20	

TABLE 3: TASK

Create the student assessments table and save it as *Assessment*.

Field name	Data type	Field size	Format	Validation rule	Description
StudentNumber	Text	13			Foreign Key
SubjectCode	Text	6			Foreign Key
TaskNumber	Number	Byte			Foreign Key
AssessmentNo	Number	Byte		1 OR 2	
AssessmentDate	Date		Medium		
AssessmentMark	Number	Integer			

TABLE 4: ASSESSMENT

Create a one-to-many relationship and enforce the referential integrity between Student and Assessment, Subject and Assessment, Task and Assessment.

HINT: See FIGURE 6.

Open the database table *Student* and add the following records:

StudentID	First Name	Surname	Date enrolled
0004045658089	Grace	Tempest	2019/02/14
9809095656088	Delina	Mabitsela	2019/12/02
9912120055087	Marvel	Biyela	2019/02/23

Open the database table *Subject* and add the following records:

Subject code	Subject name	Subject level
102305	Contact Centre Operations	L2
102306	Introduction to Systems Development	L2
102401	Computer Hardware and Software	L3

Open the database table *Task* and add the following records:

Task number	Description
100	PAT
101	Formal Test
102	Internal Exam

Copyright reserved

Open the database table *Assessment* and add the following records:

StudentID	Subject code	Task number	Assessment number	Assessment date	Assessment mark
0004045658089	102305	101	1	2019/03/23	56
0004045658089	102401	100	1	2019/03/23	65
9809095656088	102401	100	1	2019/03/23	78
9809095656088	102401	100	2	2019/06/12	45
9809095656088	102401	101	1	2019/03/28	85
9809095656088	102401	101	2	2019/12/06	64
9912120055087	102306	100	1	2019/03/23	80

3.1 Open the *Assessment* table in design view mode, select the *AssessmentNo* field and make a screenshot. Paste the screenshot into a new MS Word document, insert your EXAMINATION NUMBER and make a printout. (1)

3.2 Use the database documenter to make printouts of all database table structures. Set the options as shown in FIGURE 7. (7)

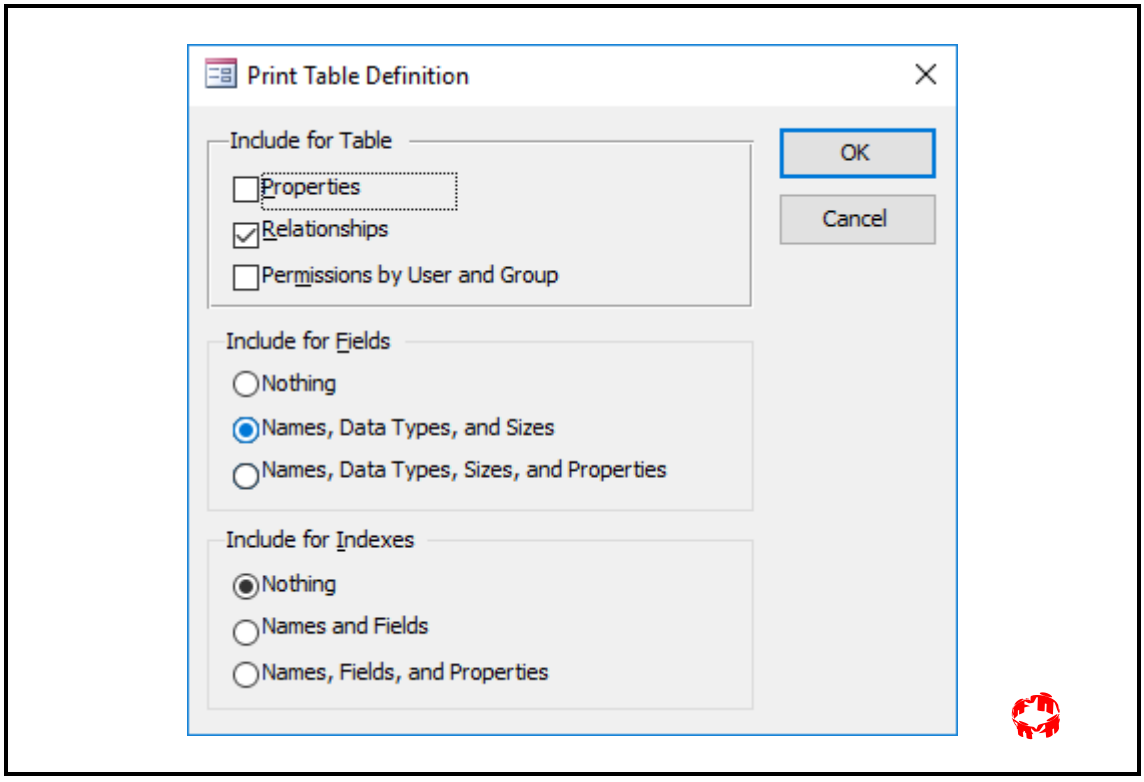


FIGURE 7

3.3 Create a database query based on all tables according to the following instructions:

3.3.1 The query must only retrieve and display data as shown in TABLE 5 where the StudentID = 0004045658089.

Table name	Field name	Instruction
Student	Full Name	Concatenate first name and surname fields with a space in the middle.
Subject	Subject Name	
Task	Description	Rename to Task
Assessment	AssessmentNo	
Assessment	AssessmentMark	

TABLE 5

(9)

3.3.2 Open the query in design view. Ensure that all column names are clearly visible. Make a screenshot and paste it into a new MS Word document. Insert your EXAMINATION NUMBER and make printout of the query design in landscape orientation.

3.4 Create a database report (like the one shown in FIGURE 8) based on the query created in QUESTION 3.3 according to the following instructions:

ICASS MARKS

SubjectName	Task	AssessmentNo	AssessmentMark
Computer Hardware & Software			
	Formal Test	1	63
	Formal Test	2	57
	PAT	1	65
		AVERAGE MARK	61,7
Contact Centre Operations			
	PAT	1	58
	Formal Test	1	56
		AVERAGE MARK	57,0

FIGURE 8

3.4.1 Show all fields from the query except *Full Name*. (2)

3.4.2 Group the report details by *Subject Name*. (1)

3.4.3 Calculate the average mark. Change its format to *Fixed* with the decimal places of 1. (3)

3.4.4 Add a label to the average calculated in QUESTION 3.4.3. (1)

3.4.5 Change the title to ICASS MARKS-EXAMINATION NUMBER (1)

3.4.6 Make a printout of the report.

[25]

TOTAL: 100

Copyright reserved