basic education

Department:
Basic Education
REPUBLIC OF SOUTH AFRICA

SENIOR CERTIFICATE/ NATIONAL SENIOR CERTIFICATE

GRADE 12

INFORMATION TECHNOLOGY P1

NOVEMBER 2020(2)

MARKS: 150

TIME: 3 hours

This question paper consists of 22 pages and 2 data pages.

INSTRUCTIONS AND INFORMATION

1. This question paper is divided into FOUR sections. Candidates must answer ALL the questions in ALL FOUR sections.

2. The duration of this examination is three hours. Because of the nature of this examination it is important to note that you will not be permitted to leave the examination room before the end of the examination session.

3. This question paper is set with programming terms that are specific to the Delphi programming language.

4. Make sure that you answer the questions according to the specifications that are given in each question. Marks will be awarded according to the set requirements.

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

6. Your programs must be coded in such a way that they will work with any data and not just the sample data supplied or any data extracts that appear in the question paper.

7. Routines, such as search, sort and selection, must be developed from first principles. You may NOT use the built-in features of Delphi for any of these routines.

8. All data structures must be defined by you, the programmer, unless the data structures are supplied.

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

10. Make sure that your examination number appears as a comment in every program that you code, as well as on every event indicated.

11. If required, print the programming code of all the programs/classes that you completed. You will be given half an hour printing time after the examination session.

12. At the end of this examination session you must hand in a disk/CD/DVD/ flash disk with all your work saved on it OR you must make sure that all your work has been saved on the disk space allocated to you for this examination session. Ensure that all files can be read.

13. The files that you need to complete this question paper have been given to you on the disk/CD/DVD/flash disk or on the disk space allocated to you. The files are provided in the form of password-protected executable files.

NOTE: Candidates must use the file **DataENGNovBackup2020.exe**.

Do the following:

- Double click on the password-protected executable file.
- Click on the extract button.
- Enter the following password: **Ent&*%ain9**

Once extracted, the following list of files will be available in the folder **DataENGNovBackup2020**:

SUPPLIED FILES:

Question 1:
HighScores.txt
Question1_P.dpr
Question1_P.dproj
Question1_P.res
Question1_U.dfm
Question1_U.pas
Rocket.bmp

Question 2:
ConnectDB_U.pas
EntertainDB.mdb
Question2_P.dpr
Question2_P.dproj
Question2_P.res
Question2_U.dfm
Question2_U.pas

Question 3:
Act_U.pas
Qualify.bmp
Question3_P.dpr
Question3_P.dproj
Question3_P.res
Question3_U.dfm
Question3_U.pas

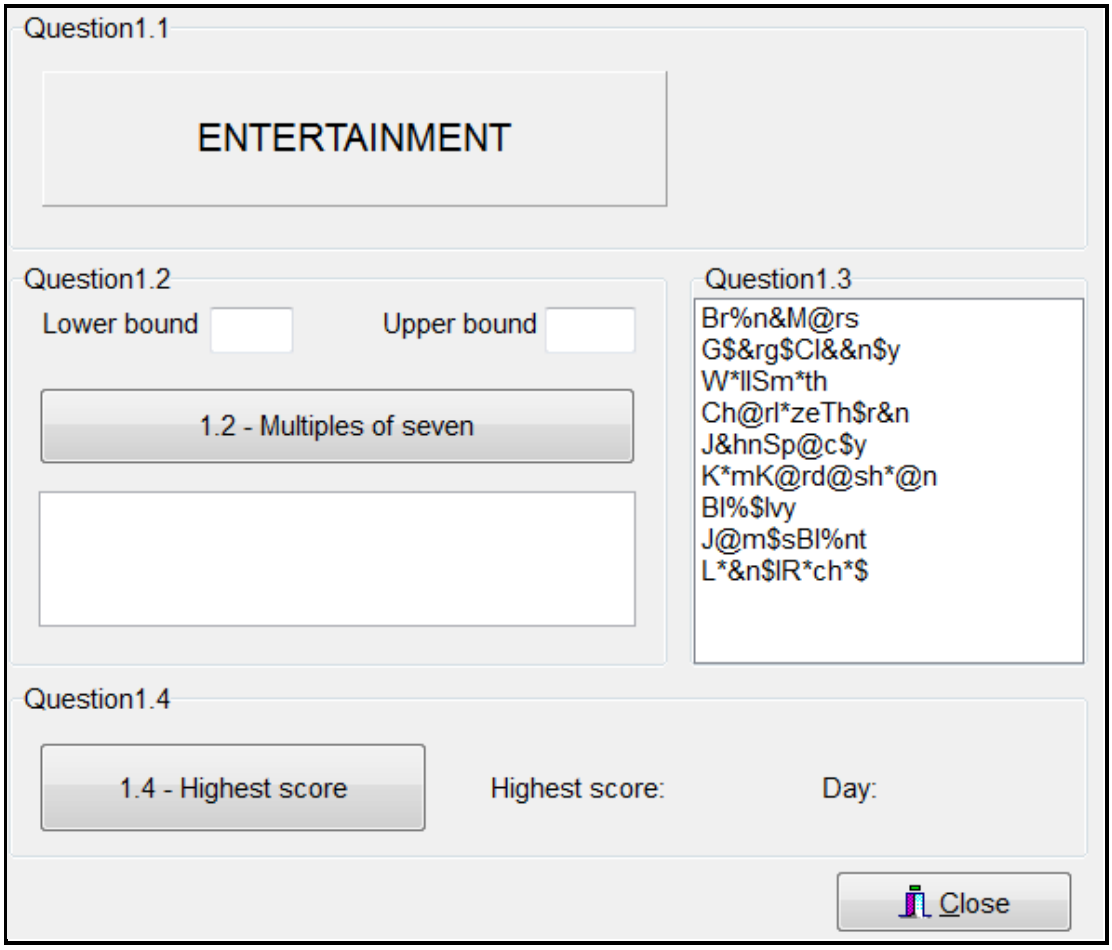
Question 4:
Question4_P.dpr
Question4_P.dproj
Question4_P.res
Question4_U.dfm
Question4_U.pas

SECTION A

QUESTION 1: GENERAL PROGRAMMING SKILLS

Do the following:

- Open the incomplete program for QUESTION 1 in the **Question 1** folder.
- Enter your examination number as a comment in the first line of the **Question1_U.pas** file.
- Compile and execute the program. The program has no functionality currently.
- The user interface used for QUESTION 1 displays four different sections called Question1.1 to Question1.4.



- Follow the instructions below to complete the code for EACH section as described in QUESTION 1.1 to QUESTION 1.4.

1.1 OnActivate event of the form

Write code to do the following:

- Change the text on the panel **pnlHeading** to be displayed in bold.
- Load the **Rocket.bmp** image so that it fits in the image component, **imgRocket**.

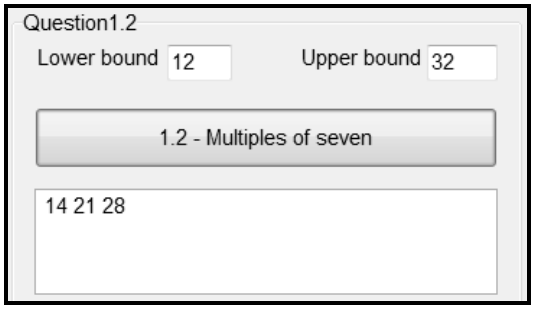
Example of output when the program is executed:

(4)

1.2 **Button [1.2 - Multiples of seven]**

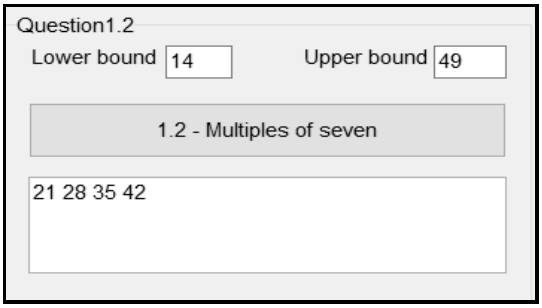
The program must determine and display the multiples of seven between a range of integer numbers (excluding the lower and upper bound numbers). The user must enter the lower and upper bounds of the range of numbers in the edit boxes provided. Display the multiples of seven in the **redQ1_2** rich edit component in one line.

Example of output if the value of 12 was entered as the lower bound and 32 was entered as the upper bound:



(12)

Example of output if the value of 14 was entered as the lower bound and 49 was entered as the upper bound:



1.3 **List box**

Some celebrities replace their names on their dressing room doors with an encrypted version. The encryption is done by removing all the spaces and replacing the vowels in their names with the special characters shown below.

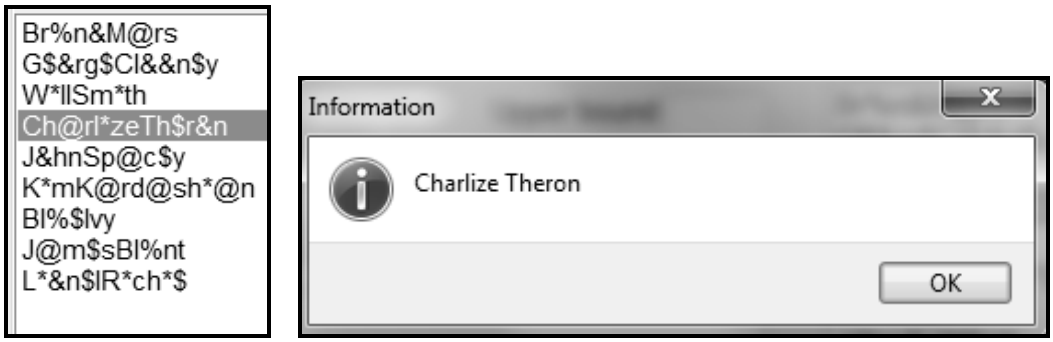
Vowel:	a	e	i	o	u
Replacement character:	@	$	*	&	%

A list box, **lstCelebrities**, contains codes that represent the names of well-known celebrities, e.g. **`Ch@rl*zeTh$r&n`**, which represents the name Charlize Theron.

The user must select a code from the list box. The program must decrypt the code and display the name in a dialogue box. The decryption must be done as follows:

- Replace all the special characters with the corresponding vowels.
- Each code starts with a capital letter. The next capital letter is that of the surname. Insert a space before the capital letter for the surname in the code.
- Display the decrypted name accordingly.

Example of output if the code **`'Ch@rl*zeTh$r&n'`** was selected:

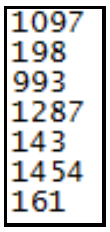


(12)

1.4 **Button [1.4 - Highest score]**

A gamer saved the high scores that he achieved playing his favourite online game in a text file called **HighScores.txt**. The data is saved one score per line for each day of the week starting at day 1.

Example of the contents of the text file:



Write code to do the following:

- Read the data from the text file and determine the highest score achieved, as well as the number of the day of the week when this score was achieved.
- Display the highest score and the number of the day it was achieved on the label components provided.

NOTE: Hardcoding will NOT be accepted as the program must work for any set of data contained in the text file.

Example of output:

(12)

- Ensure that your examination number has been entered as a comment in the first line of the program file.
- Save your program.
- Print the code if required.

TOTAL SECTION A: 40

SECTION B

QUESTION 2: DATABASE PROGRAMMING

A school is organising a cultural festival which will be held from 12 to 18 April 2021. A variety of productions will be staged and tickets for the productions may be bought during the festival.

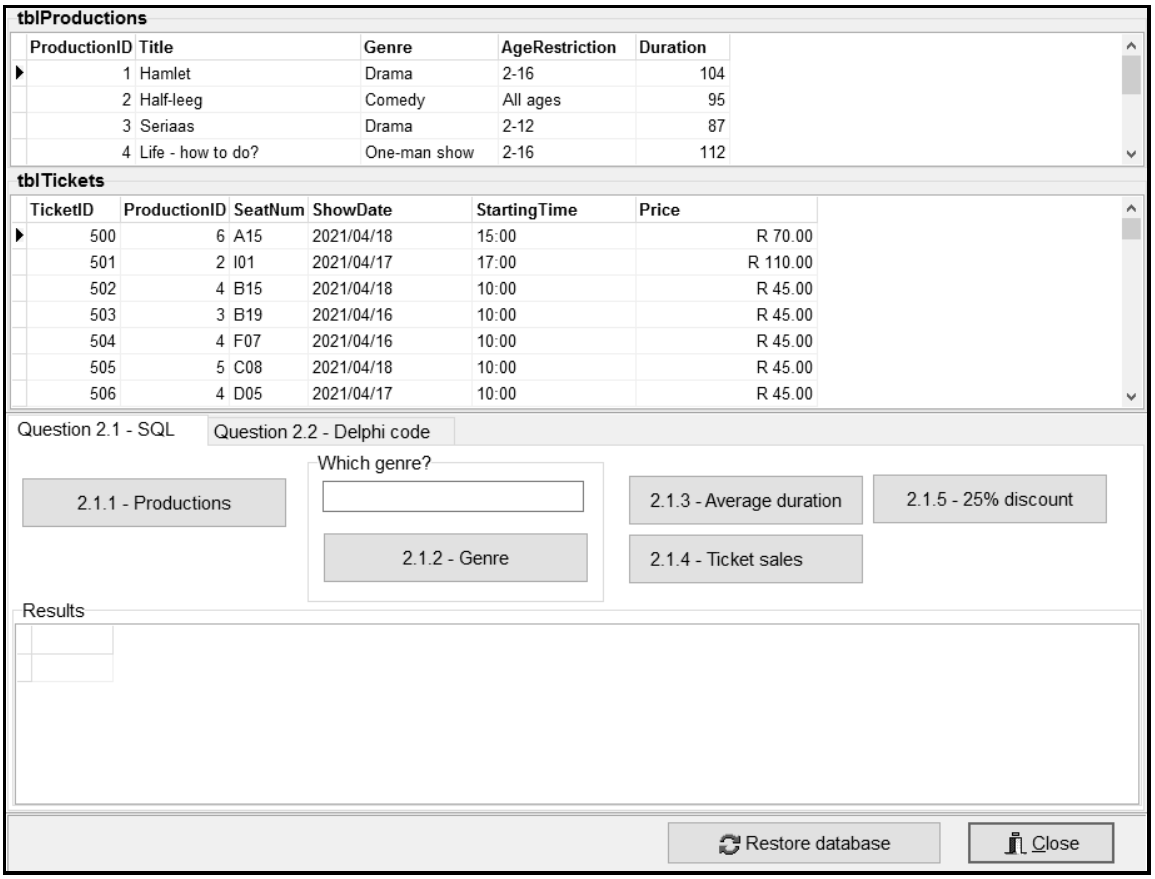
A database called **EntertainDB** and an incomplete project called **Question2_P** are provided in the **Question 2** folder. The database contains two tables, namely **tblProductions** and **tblTickets** with data related to the festival that will be held during April 2021.

The data pages at the end of the question paper provide information on the design of the database and its contents.

Do the following:

- Open the incomplete project file called **Question2_P.dpr** in the **Question 2** folder.
- Enter your examination number as a comment in the first line of the **Question2_U.pas** file.
- Compile and execute the program. The program has no functionality currently.

The following user interface is displayed:



- Follow the instructions below to complete the code for each section, as described in QUESTION 2.1 and QUESTION 2.2.
- Use SQL code to answer QUESTION 2.1 and Delphi code to answer QUESTION 2.2.

NOTE:

- The **[Restore database]** button is provided to restore the data contained in the database to the original content. If you need to test your code on the original data, you may click this button to restore the data.
- Code is provided to link the GUI components to the database.
- Do NOT change any of the code provided.
- TWO variables are declared as public variables, as described in the table below:

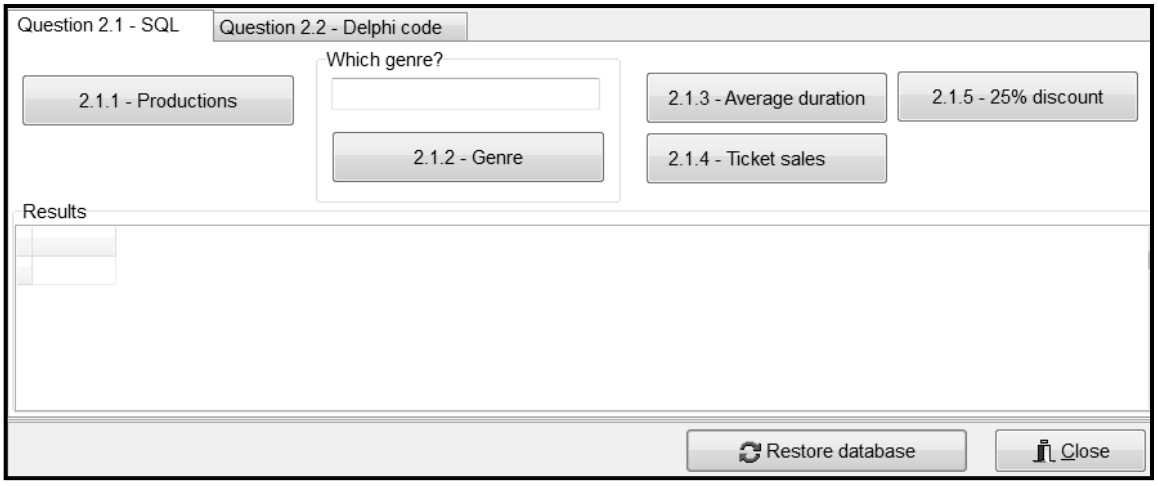
Variable	Data type	Description
`tblProductions`	`TADOTable`	Refers to the table **tblProductions**
`tblTickets`	`TADOTable`	Refers to the table **tblTickets**

2.1 Tab sheet [Question 2.1 - SQL]

You may use ONLY SQL statements in this section to answer QUESTION 2.1.1 to QUESTION 2.1.5.

Code to execute the SQL statements and display the results of the queries is provided. The SQL statements are incomplete.

The following user interface is displayed:



Do the following:

Complete the SQL statement for the given string variable named **sSQL** in each of the questions.

2.1.1 **Button [2.1.1 - Productions]**

Display ALL the fields from the **tblProductions** table, sorted according to duration in ascending order.

Example of output for the first five records:

Results

ProductionID	Title	Genre	AgeRestriction	Duration
3	Seriaas	Drama	2-12	87
2	Half-leeg	Comedy	All ages	95
5	Mike and Mavis	Drama	All ages	97
6	When in doubt…	Comedy	2-12	103
1	Hamlet	Drama	2-16	104

(3)

2.1.2 **Button [2.1.2 - Genre]**

The user must enter the name of a genre in the **edtGenre** edit box provided. Display the title and age restriction of the productions of the genre that has been entered.

Example of output if the genre entered is 'Drama':

Results

Title	AgeRestriction
Hamlet	2-16
Seriaas	2-12
Mike and Mavis	All ages

(4)

2.1.3 **Button [2.1.3 - Average duration]**

Display the average duration of all productions in the **tblProductions** table, rounded to THREE decimal digits. Display the calculated field with the heading '**AverageDuration**'.

Example of output:

Results

AverageDuration
99.667

(4)

2.1.4 **Button [2.1.4 - Ticket sales]**

Display the title and the total number of tickets sold for each production. Use the heading '**Number of tickets sold**' for the calculated field.

Example of output for the first five records:

Results

Title	Number of tickets sold
Half-leeg	37
Hamlet	44
Life - how to do?	34
Mike and Mavis	35
Seriaas	49

(7)

2.1.5 **Button [2.1.5 - 25% discount]**

The festival organisers declared that all tickets for seats in rows C, D, E and F for any production during the show that starts at 10:00 must receive a 25% discount.

Complete the code to update the database to reflect the discounted prices.

NOTE: Indicate the time as 10.00 instead of 10:00, because using time in SQL statements in Delphi requires the substitution of the colon ('**:**') with a full stop ('**.**').

Example of output of the first five records:

Results

TicketID	ProductionID	SeatNum	ShowDate	Starting Time	Price
500	6	A15	2021/04/18	15:00	R 70.00
501	2	I01	2021/04/17	17:00	R 110.00
502	4	B15	2021/04/18	10:00	R 45.00
503	3	B19	2021/04/16	10:00	R 45.00
504	4	F07	2021/04/16	10:00	R 33.75

NOTE: The date format may differ from the screenshot.

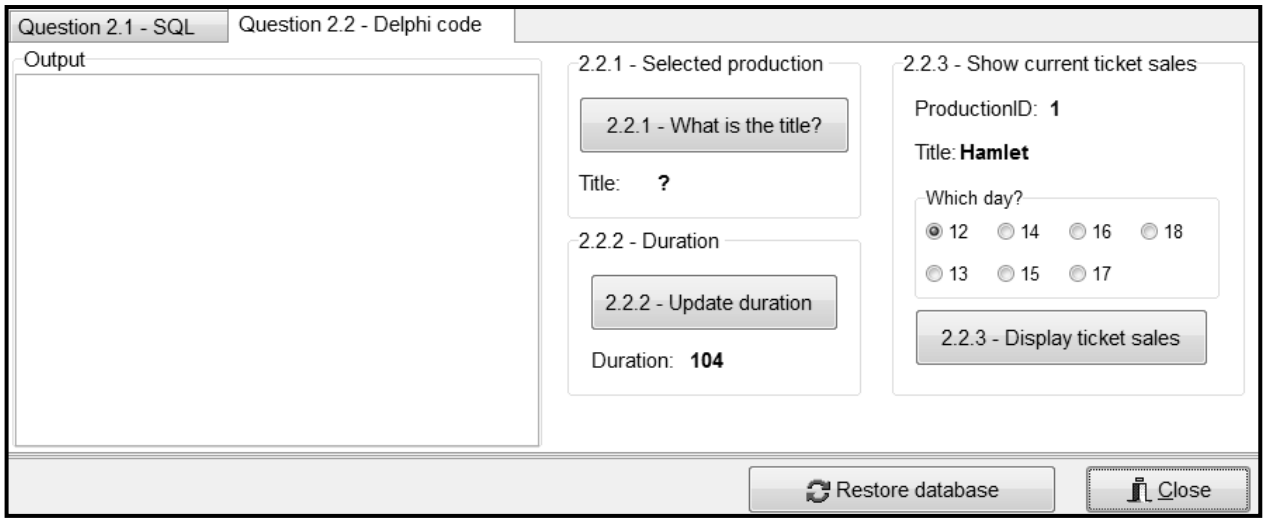
(7)

2.2 **Tab sheet [Question 2.2 - Delphi code]**

In this section only Delphi programming code may be used to answer QUESTION 2.2.1 to QUESTION 2.2.3.

NO marks will be awarded for SQL statements in QUESTION 2.2.

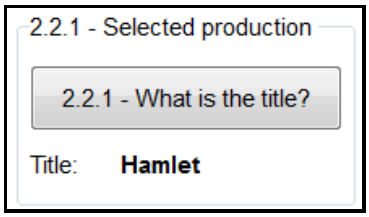
The user interface for QUESTION 2.2 is shown on the next page.



2.2.1 **Button [2.2.1 - What is the title?]**

Write code to replace the question mark ('**?**') in the label **lblTitle** to display the title of the currently selected production in the **tblProductions** table.

Example of output if the first record is currently selected in the **tblProductions** table:



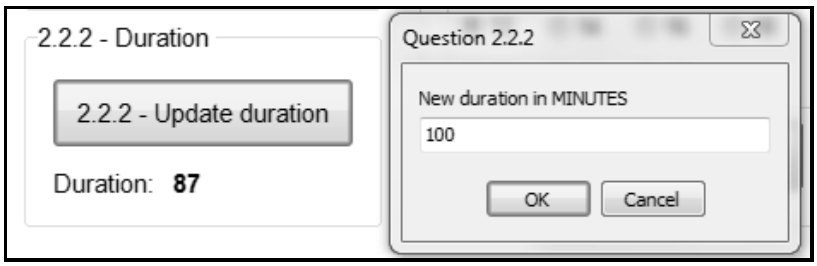
(2)

2.2.2 **Button [2.2.2 - Update duration]**

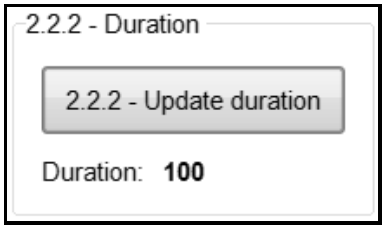
The user interface automatically displays the duration of the production currently selected in the **tblProductions** table. When the user clicks the button, provided code will request the user to enter the new duration for the production in minutes using an input box.

Write code to change the duration of the production record currently selected in the **tblProductions** table to the value that has been entered.

Example of input if the duration for the production *Seriaas* must be 100 minutes instead of 87 minutes:



Example of output after the duration of the selected production was changed:



(3)

2.2.3 **Button [2.2.3 - Display ticket sales]**

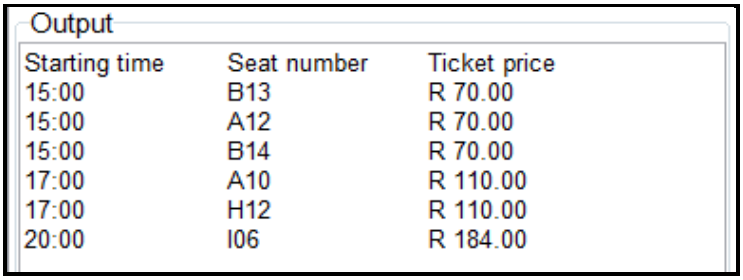
The festival manager needs a report to show all the tickets sold for a production on a specific day. The user must select the specific day in April from the radio group named **rgpWhichDay**.

Write code to display the starting time, the seat number booked and the price of each ticket sold for the selected production on the selected day.

NOTE:

- Code has been provided to display the headings.
- The information must be displayed in neat columns below the headings.
- The prices of the tickets must be displayed in currency format.

Example of output if the production *Hamlet* was selected for 12 April 2021:



Example of output if the production *Half-leeg* was selected for 16 April 2021:

Output

Starting time	Seat number	Ticket price
10:00	E03	R 45.00
10:00	C07	R 45.00
15:00	C23	R 70.00
17:00	H08	R 110.00

(10)

- Ensure that your examination number has been entered as a comment in the first line of the program file.
- Save your program.
- Print the code if required.

TOTAL SECTION B: 40

SECTION C

QUESTION 3: OBJECT-ORIENTATED PROGRAMMING

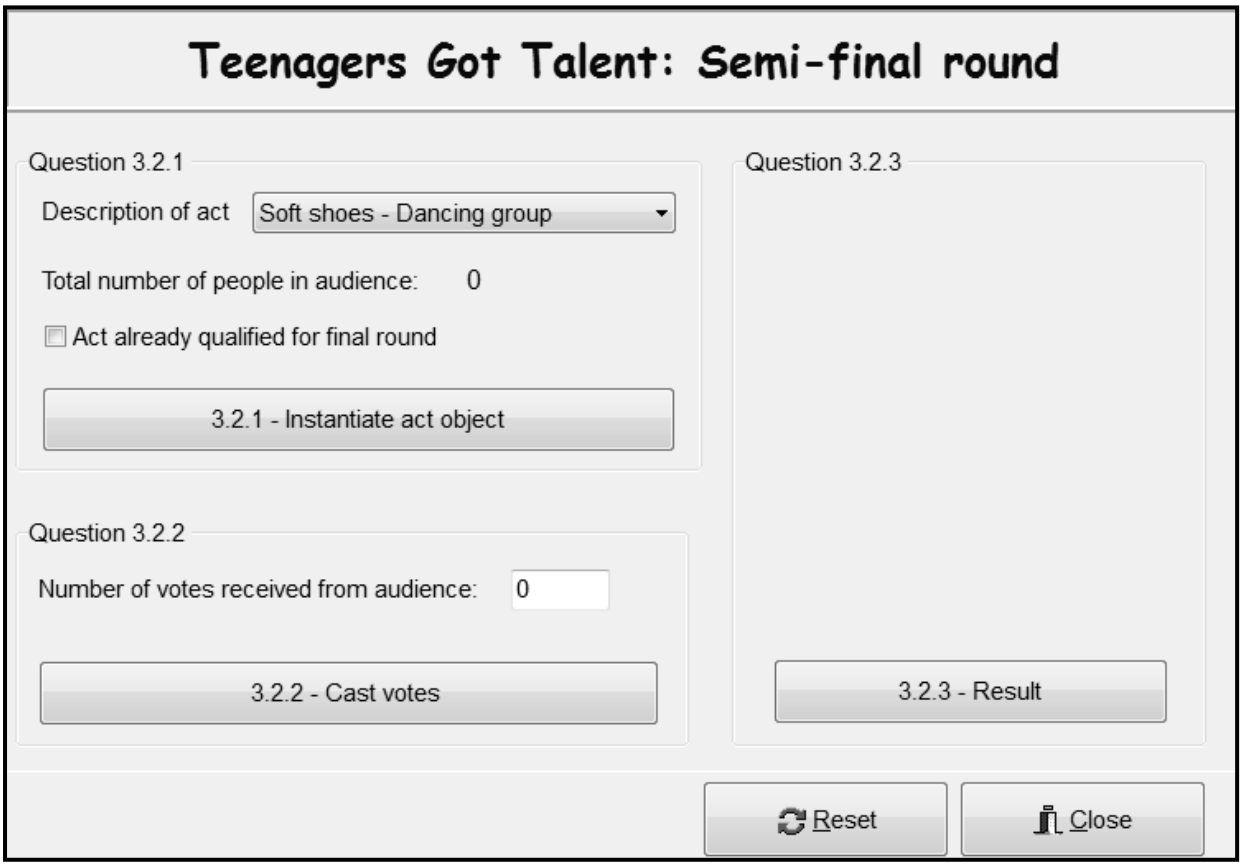
The results for the semi-final round for the Teenagers Got Talent show must be captured to determine which acts qualify for the final round.

An act qualifies for the final round if more than 600 members of the audience voted for the act or if the act received a special vote from the judges, which means that it automatically qualifies for the final round.

Do the following:

- Open the incomplete program in the folder **Question 3**.
- Open the incomplete object class **Act_U.pas**.
- Enter your examination number as a comment in the first line of both the **Question3_U.pas** file and the **Act_U.pas** file.
- Compile and execute the program. The program has no functionality currently.

The following user interface is displayed:



- Complete the code as specified in QUESTION 3.1 for the **Act_U** object class and in QUESTION 3.2 for the **Question3_U** form class.

3.1 The incomplete object class (**TAct**) provided contains the following:

- Declarations of four attributes that define an **Act** object
- A completed **toString** method to display the attributes

The attributes for the **Act** object have been declared as follows:

Names of attributes	Description
fActDescription	Description of the act
fPreQualify	A Boolean value of 'true' indicating that the act prequalified for the final round or 'false' if not
fAudienceSize	The total number of people in the audience attending the semi-final round
fAudienceVotes	The number of votes received from members of the audience

3.1.1 Write code for a constructor method that will receive the description of the act, a Boolean value indicating whether the act prequalified for the final round or not and the total number of members in the audience as parameter values. Assign these values to the respective attributes.

Set the number of votes received from the audience to zero. (5)

3.1.2 Write code for an accessor method called **getActDescription** for the attribute that describes the act. (3)

3.1.3 Write code for a mutator method called **setAudienceVotes** that receives the number of audience members who voted for the act and updates the relevant attribute. (3)

3.1.4 Write code for a method called **checkStatus** that will return a Boolean value that indicates whether the act qualifies for the final round or not.

An act qualifies for the final round if any one of the following requirements are met:

- More than 600 members of the audience voted for the act.
- The act already qualified for the final round. (7)

3.2 An incomplete unit **Question3_U** has been provided and contains code for the object class to be accessible.

The following code is provided:

- A globally declared object variable **objAct**
- A globally declared variable called **iAudienceSize**. A randomly generated value in the range 700 to 800 is assigned to **iAudienceSize** in the **OnChange** event of the combo box **cmbActs**.
- Code in the [**Reset**] button that will restore the properties of the components to its original state

Do the following to complete the code for QUESTION 3.2.1 to QUESTION 3.2.3 in the main form unit:

3.2.1 **Button [3.2.1 - Instantiate act object]**

The user must do the following:

- Choose the description of the act from the combo box.
- Select the **chkPreQualified** check box to indicate whether or not the act already qualified for the final round.

Write code to do the following:

- Use the description of the act, the audience size and the Boolean value from the check box indicating whether the act prequalified or not, and instantiate an **Act** object.
- Display a message to indicate that the object has been instantiated. (7)

3.2.2 **Button [3.2.2 - Cast votes]**

Write code to do the following:

- Extract the number of votes received from the audience for the act.
- If the number of votes received from the audience is more than the total size of the audience, then display a message 'Invalid' and return the focus to the edit box **edtAudienceVotes**.
- If the number of votes is less or equal to the total size of the audience, then use the **setAudienceVotes** method to set the value of the attribute of the object. (6)

3.2.3 **Button [3.2.3 - Result]**

Write code to determine whether the act qualifies for the final round using the **checkStatus** method.

NOTE: The panels, **pnlQualify** and **pnlNotQualify**, are the same size and placed in the exact same position on the GUI. Therefore, the one panel is hidden behind the other panel.

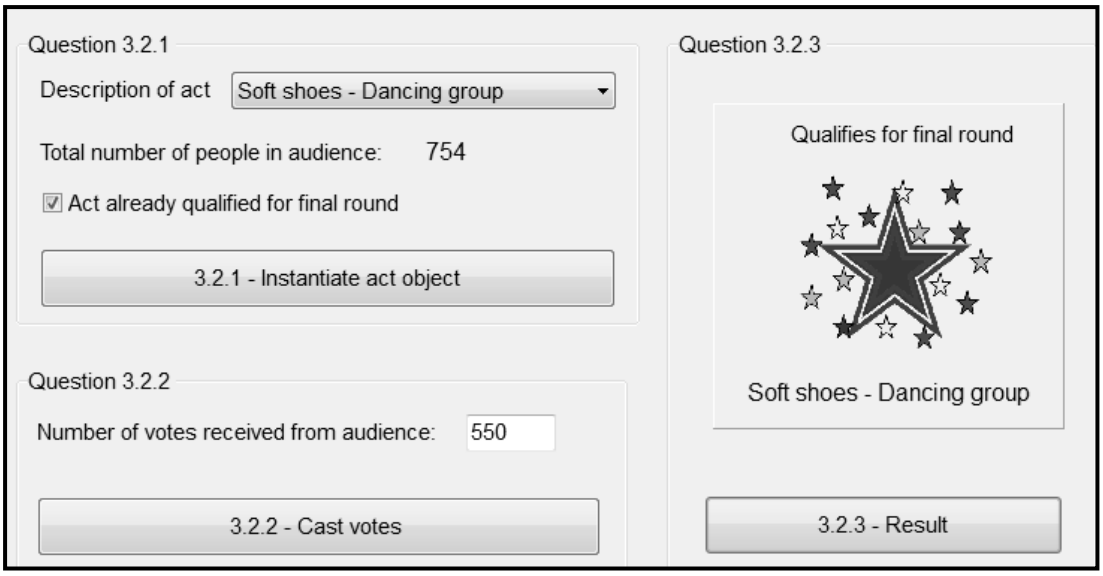
If an act qualifies for the final round:

- Set the panel **pnlQualify** to be visible and hide the panel **pnlNotQualify**.
- Use the relevant accessor method to get the description of the act and display it on the label **lblQualify**.

If an act does NOT qualify for the final round:

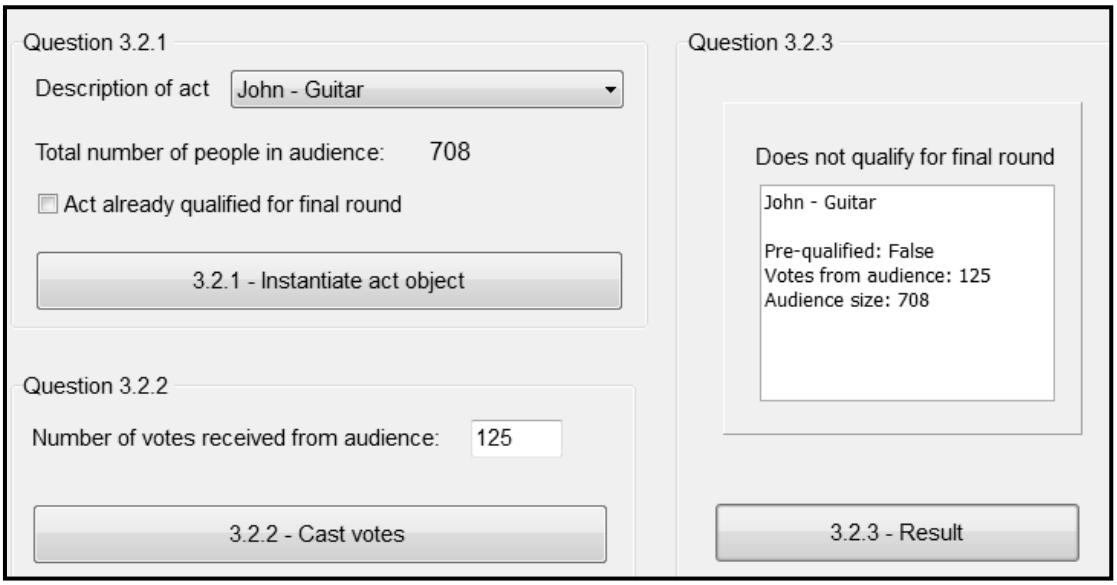
- Set the panel **pnlNotQualify** to be visible and hide the panel **pnlQualify**.
- Display the information of the object in the **redNotQualify** output area using the **toString** method.

Example of input and output if an act qualified for the final round:



NOTE: The total number of people in the audience is randomly generated and may differ from the number shown on the user interface when you execute the program.

Example of input and output if an act does NOT qualify for the final round:



(9)

- Ensure that your examination number has been entered as a comment in the first line of the object class and the form class.
- Save all the files.
- Print the code if required.

TOTAL SECTION C: 40

SECTION D

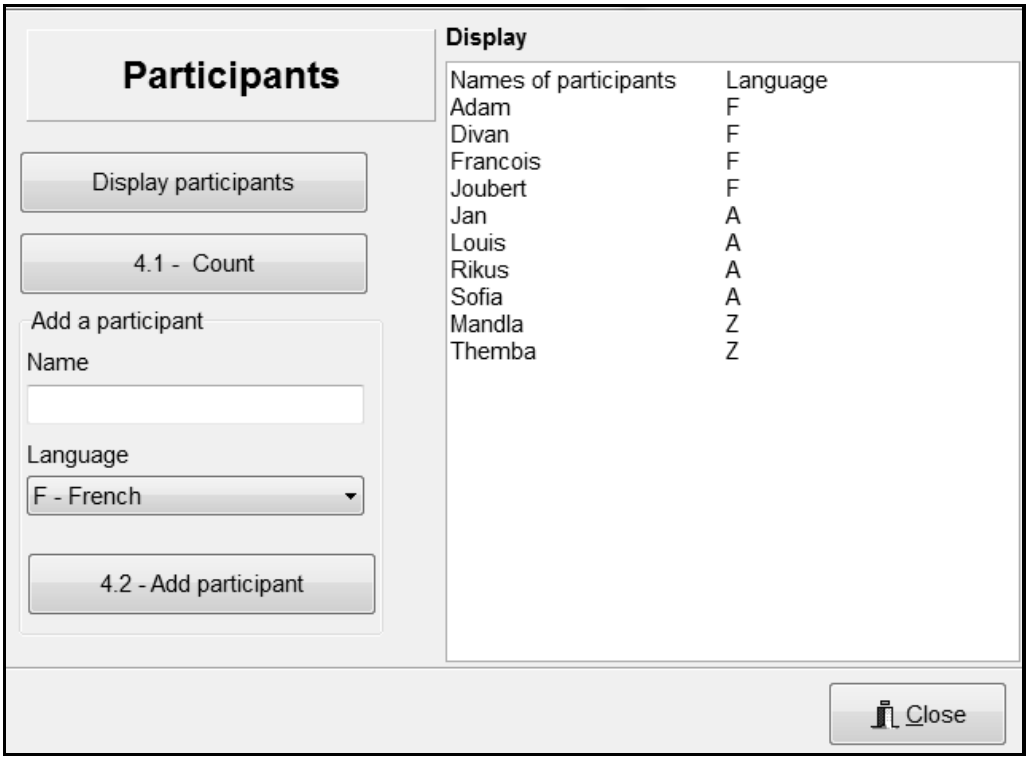
QUESTION 4: PROBLEM-SOLVING PROGRAMMING

The French Embassy is offering students an opportunity to participate in a cultural event. The stage names and languages of all participants are stored in two parallel arrays. The manager must be able to add the information of new participants at the event.

Do the following:

- Open the incomplete program in the **Question 4** folder.
- Enter your examination number as a comment in the first line of the **Question4_U.pas** file.
- Compile and execute the program. The program currently has limited functionality.

  The following user interface is displayed:



Code has been provided for the declaration of the following **global variables**, the **OnCreate** event handler and the **btnDisplay** button.

The following have been provided:

- **arrLanguages**, which contains the names of the three languages that may be used at the cultural event:

```
arrLanguages: array[1..3] of String = ('French', 'Afrikaans',
'Zulu');
```

- **arrNames** that can store a maximum of 100 names of participants:

```
arrNames: array[1..100] of String;
```

- **arrLangCode** that can store a maximum of 100 codes. A code is the first letter of the language that the participant has chosen, e.g. the language 'French' will be represented by the character 'F' in the **arrLangCode** array.

  NOTE: The language isiZulu is saved as Zulu in the **arrLanguage** array and is represented by the character 'Z'.

```
arrLangCode: array[1..100] of char;
```

- A variable **iSize** to keep track of the current number of participants in the **arrNames** array:

```
iSize: integer;
```

- The **OnCreate** event handler contains code to set the **iSize** variable to ten and to populate parallel arrays **arrNames** and **arrLangCode** with the names and codes of the current ten participants.
- The '**Display participants**' button contains code to display the current data stored in the arrays **arrNames** and **arrLangCode**.
- The data in the **arrNames** array has been sorted to display the names of the participants in alphabetical order, grouped by their language.
- The language groups must always start with French, followed by Afrikaans and then Zulu.

Example of output for the **onClick** event of the **btnDisplay** button:

Display

Names of participants	Language
Adam	F
Divan	F
Francois	F
Joubert	F
Jan	A
Louis	A
Rikus	A
Sofia	A
Mandla	Z
Themba	Z

Write code to perform the tasks described in QUESTION 4.1 and QUESTION 4.2 below.

4.1 **Button [4.1 - Count]**

Calculate and display the total number of participants per language.

Example of output for the original data in the arrays:

Display

Language	Number of participants
French	4
Afrikaans	4
Zulu	2

(11)

4.2 **Button [4.2 - Add participant]**

The information of a new participant must be added by entering the name in the **edtName** edit box and selecting the language in the **cmbLanguage** combo box.

Code has been provided to do the following:

- Extract the name from the edit box **edtName** and convert the name into the correct format for it to be added to the array. The format of the names in the array has the first letter in uppercase and all the other letters in lowercase, e.g. Sam.
- Extract the first letter of the language selected from the combo box **cmbLanguage**.

Write code to do the following:

- Insert the name of the participant extracted from the **edtName** edit box to the correct position alphabetically in the sorted array **arrNames**.
- Insert the first letter of the language of the participant in the correct position in array **arrLangCode**.

NOTE: The language group order must always be French, Afrikaans and then Zulu. Inserting a new participant and language code should NOT change the alphabetical and/or language group order.

Example of output if the name entered is Garner and the language selected is French:

Display

Names of participants	Language
Adam	F
Divan	F
Francois	F
Garner	F
Joubert	F
Jan	A
Louis	A
Rikus	A
Sofia	A
Mandla	Z
Themba	Z

Example of output if the next name entered is Tessa and the language selected is Afrikaans:

Display

Names of participants	Language
Adam	F
Divan	F
Francois	F
Garner	F
Joubert	F
Jan	A
Louis	A
Rikus	A
Sofia	A
Tessa	A
Mandla	Z
Themba	Z

(19)

- Ensure that your examination number has been entered as a comment in the first line of the program file.
- Save your program.
- Print the code if required.

TOTAL SECTION D: 30
GRAND TOTAL: 150

INFORMATION TECHNOLOGY P1(2)

QUESTION 2: DATABASE INFORMATION

The design of the database tables is as follows:

Table: **tblProductions**

This table contains the data for the various productions that will be performed at the festival.

Field name	Data type	Size	Description
ProductionID	Number	Long Integer	Unique number assigned to each production
Title	Text	25	The title of the production, e.g. *Half-leeg*
Genre	Text	15	The genre of the production, e.g. Drama, Comedy, etc.
AgeRestriction	Text	15	The age restriction imposed on the production, e.g. 2-16, All ages
Duration	Number	Integer	The duration of the production in minutes, e.g. 112

Example of data:

tblProductions

ProductionID	Title	Genre	AgeRestriction	Duration
1	Hamlet	Drama	2-16	104
2	Half-leeg	Comedy	All ages	95
3	Seriaas	Drama	2-12	87
4	Life - how to do?	One-man show	2-16	112

Table: **tblTickets**

This table contains the data on the tickets currently sold to attend the various productions.

Field name	Data type	Size	Description
TicketID	Number	Long Integer	Unique number assigned to each ticket bought
ProductionID	Number	Long Integer	A number assigned to each production
SeatNum	Text	5	The number of the seat the ticket was bought for, e.g. B2 – which indicates that the seat is in row B, seat number 2
ShowDate	DateTime	ShortDate	The date when the production will be attended during the week of 12 to 18 April 2021
StartingTime	DateTime	ShortTime	The time slot the production will be attended. The possible options are: 10:00, 15:00, 17:00 and 20:00.
Price	Currency		The price of the ticket

Example of data of the first seven records:

tblTickets

TicketID	ProductionID	SeatNum	ShowDate	StartingTime	Price
500	6	A15	2021/04/18	15:00	R 70.00
501	2	I01	2021/04/17	17:00	R 110.00
502	4	B15	2021/04/18	10:00	R 45.00
503	3	B19	2021/04/16	10:00	R 45.00
504	4	F07	2021/04/16	10:00	R 45.00
505	5	C08	2021/04/18	10:00	R 45.00
506	4	D05	2021/04/17	10:00	R 45.00

NOTE: The date format may differ from the screenshot.

The following one-to-many relationship with referential integrity exists between the two tables in the database: