NATIONAL SENIOR CERTIFICATE EXAMINATION
NOVEMBER 2024

INFORMATION TECHNOLOGY: PAPER I

Time: 3 hours 150 marks

PLEASE READ THE FOLLOWING INSTRUCTIONS CAREFULLY

1. This question paper consists of 16 pages. Please check that your question paper is complete.

2. This question paper is to be answered using object-oriented programming principles. Your program must make sensible use of methods and parameters.

3. This paper is divided into two sections. All candidates must answer both sections.

4. This paper is set in programming terms that are not specific to any particular programming language (Java/Delphi) or database (Access/MySQL/JavaDB).

5. Ensure that you answer the questions in the manner described because marks will be awarded for your solution according to the specifications that are given in the question.

6. Only answer what is asked for in each question. For example, if the question does not ask for data validation, then no marks are awarded for it; therefore, no code needs to be written for data validation.

7. If you cannot get a section of code to work, comment it out so that it will not be executed, and you can continue with the examination. If possible, try to explain the error to aid the marker.

8. Your programs must be coded so that they will work with any data, not just the sample data supplied or any data extracts that appear in the question paper. You are advised to look at the provided data files carefully.

9. Make sure that routines such as searches and sorts for arrays are developed from first principles and that you do not use the built-in features of a programming language for any of these routines.

10. All data structures must be defined and declared by you, the programmer. You may not use the interface's components to store and retrieve data.

11. Read the whole question paper before you choose a data structure. You may find that there could be an alternative method of representing the data that will be more efficient considering the questions that are asked in the paper.

12. You must save all your work regularly on the disk you have been given, or the disk space allocated to you for this examination. You should also create a backup of the original files before you start in case your solution accidentally modifies the original version.

13. If your examination is interrupted by a technical problem such as a power failure, you will, when you resume writing, be given only the time that was remaining when the interruption began, to complete your examination. No extra time will be given to catch up on work that was not saved.

14. Make sure that your examination number appears as a comment in every program you code and on every printed page you hand in.

15. Check every printed page you print, making sure you submit the code for every question.

16. Check that your printout has correct indenting and formatting and has not truncated any text.

17. Check that each page is clearly legible with a font size of 12.

18. Print a code listing of all the programs/class files that you code. Printing must be done after the examination. You will be given half an hour to print after the examination is finished. Your teacher will tell you what arrangements have been made to print your work.

19. You should have the following two folders (in bold) and files. These files are to be used as data for this examination. Note that the database files are in MS Access, JavaDB and MySQL format and contain the same data in each file. You only need the file that matches the database application you will need for Section A. Ensure that you can open the files with the packages you will use to code your solutions to this examination.

    Section A:
    SAMusic_Access.mdb
    SAMusic_JavaDB.sql
    SAMusic_MySQL.sql
    SQLAnswerSheet.rtf
    SQLBrowser.exe

    Section B:
    MusicData.txt

SECTION A SQL

SCENARIO:

QUESTION 1

South African Records manages some solo musicians and bands. The musicians and bands perform at festivals throughout the year. All the information on musicians, festivals and bookings is contained in the database **SAMusic**.

Some festivals are repeated on different dates (for example the Valentine Fire festival was held in 2022, 2023 and 2024).

tblMusicians contains the details of all the solo musicians and bands. Each row is either a solo musician or a band, determined by the value in the Solo column.

FIELDS	DATA TYPE	DESCRIPTION
MusID	INTEGER	A unique auto-numbered identification number for each musician or band.
StageName	TEXT	The name used by the musician or band for performances.
Fee	INTEGER	The amount of money the musician or band charges for a performance.
Province	TEXT	The province the musician or band is from.
Solo	BOOLEAN	Whether the musician or band is a solo performer.

The records in tblMusicians:

MusID	StageName	Fee	Province	Solo
1	Rebel	174651	Gauteng	No
2	Thabisile Ndlovu	188852	Mpumalanga	Yes
3	Flowers	127890	Western Cape	No
4	Samuel Ericson	80732	Gauteng	Yes
5	Easy K	167545	Gauteng	Yes
6	Kusuma Chapman	81282	Western Cape	Yes
7	Roger Hatfield	213307	Limpopo	Yes
8	Sound of Fury	186406	Mpumalanga	No
9	Springheeled Dassies	91651	Gauteng	No

tblFestivals contains the details of the festivals that the musicians and bands performed at.

FIELDS	DATA TYPE	DESCRIPTION
FestID	INTEGER	A unique auto-numbered identification number for each festival.
FestName	TEXT	The name of the festival.
FestDate	DATE/TIME	The day the festival occurs.
Capacity	INTEGER	The maximum number of people who can attend the festival.
TicketCost	INTEGER	The price charged for a ticket to the festival.

The first 10 records of tblFestivals:

FestID	FestName	FestDate	Capacity	TicketCost
1	Valentine Fire	2022/02/20	27000	200
2	Dance Into the Evening	2022/06/04	17000	180
3	Fire in June	2022/06/10	25000	200
4	Dance Into the Evening	2022/07/14	26000	140
5	In the Veldt	2022/10/05	14000	160
6	Coffee and Cake	2022/10/28	23000	290
7	Dance Into the Evening	2022/11/16	12000	140
8	Holiday Fire	2022/12/10	28000	250
9	Amapiano All the Way	2022/12/25	22000	250
10	Valentine Fire	2023/02/15	27000	220

tblBookings contains the details of all bookings made for musicians and bands to perform at a festival.

FIELDS	DATA TYPE	DESCRIPTION
FestID	INTEGER	The ID of the festival. This is a foreign key to tblFestivals.
MusID	INTEGER	The ID of the musician or band. This is a foreign key to tblMusicians.
SetTime	DATE/TIME	The time the musician or band will start performing at the festival.

The first 10 records of tblBookings:

FestID	MusID	SetTime
1	1	23:00:00
1	8	13:45:00
1	9	17:30:00
2	6	21:00:00
3	2	19:00:00
3	5	22:15:00
3	7	20:00:00
4	2	22:15:00
4	8	13:00:00
5	7	14:30:00

Note: Do not rename columns/use an alias unless explicitly told to do so in a question.

1.1 Display the details of the musicians who are solo and from Gauteng. The correct output is shown below:

MusID	StageName	Fee	Province	Solo
4	Samuel Ericson	80732	Gauteng	Yes
5	Easy K	167545	Gauteng	Yes

(4)

1.2 Display the festival name, festival date and income for all festivals. Calculate the income by multiplying the festival's capacity by the ticket cost. Name this column 'Income'.

Note only the first six records are shown.

FestName	FestDate	Income
Valentine Fire	2022/02/20	5400000
Dance Into the Evening	2022/06/04	3060000
Fire in June	2022/06/10	5000000
Dance Into the Evening	2022/07/14	3640000
In the Veldt	2022/10/05	2240000
Coffee and Cake	2022/10/28	6670000

(4)

1.3 Display the average ticket cost for all festivals whose capacity is less than 10000. The correct output is shown below:

AveCost
170

(3)

1.4 Some festivals take place on many different dates. Display the festival name and number of times the festival has occurred. Display only those festivals that occur more than once, and which have taken place before the current date. Order the data by festival name. The correct output is shown below:

FestName	NumFests
Amapiano All the Way	3
Coffee and Cake	2
Dance Into the Evening	5
Fire in June	3
Holiday Fire	2
In the Veldt	3
Valentine Fire	3

(7)

1.5 Display the first 3 characters of the stage name of all musicians that have not been booked to play at any festivals. The correct output is shown below:

StageName
Sam

(6)

1.6 Display the festival name, festival date and set times for each booking. Combine the date and time into one column, with a space between the date and time as shown in the sample output below:

Note only the first 6 rows are shown.

FestName	SetDateAndTime
Valentine Fire	2022/02/20 23:00:00
Valentine Fire	2022/02/20 13:45:00
Valentine Fire	2022/02/20 17:30:00
Dance Into the Evening	2022/06/04 21:00:00
Fire in June	2022/06/10 19:00:00
Fire in June	2022/06/10 22:15:00

(6)

1.7 South African Records wants to know the total amount they need to pay their musicians and bands for each festival. Display the festival name, festival date and total fee for the musicians and bands for each festival and for each date. Sort the result from the most recent festival to the least recent festival. The sample output is shown below:

Note only the first six records are shown.

FestName	FestDate	TotalFee
Holiday Fire	2024/12/19	526884
Durban Jol	2024/11/15	127890
Fire in June	2024/06/22	569704
Amapiano All the Way	2024/06/21	248827
In the Veldt	2024/05/19	209172
Coffee and Cake	2024/05/04	188852

(8)

1.8 The Coffee and Cake festivals will now be free of charge. Change the TicketCost to 0 for all festivals named Coffee and Cake. (4)

1.9 The Fire festivals are festivals with the word Fire anywhere in their name. The Fire festivals from 2024 will be repeated in 2025.

- The festival names will be the same.
- The festival dates will be set to 1 January 2025.
- The capacity will be the same as the 2024 festival of the same name.
- The ticket cost will be R20 more expensive than the current ticket cost in 2024 for the festival of the same name.

Write a single query to insert the Fire festivals for 2025.

The inserted records are shown below:

Note only your FestIDs may be different.

FestID	FestName	FestDate	Capacity	TicketCost
25	Valentine Fire	2025/01/01	27000	260
26	Fire in June	2025/01/01	25000	260
27	Holiday Fire	2025/01/01	28000	310

(8)

50 marks

SECTION B OBJECT-ORIENTATED PROGRAMMING

South African Records is a record label that has contracts with some South African musicians. The company manages the songs that musicians create while they are contracted. Musicians will have a different number of songs. Some musicians may not have created any songs since the start of their contract. Songs can be liked and disliked by listeners. The number of likes and dislikes is stored. Some musicians have stage names that are different to their full names.

The four different genres of songs created by the musicians are represented by numbers as follows:

- 1 – Rock
- 2 – Gqom
- 3 – Amapiano
- 4 – Techno

South African Records needs to create a list of easy listening songs that are suitable for playing on any radio station. An easy listening song is any song that has more than 5000 likes and more likes than dislikes.

Musician Class

This class will represent the basic details of musicians.

- **name** – the full name of the musician, stored as a string.
- **stageName** – the stage name of the musician, stored as a string.
- **genre** – the genre of music the musician creates, stored as a number.
- **start** – the date the musician started their contract with South African Records, stored as a Date in the format **yyyy<space>MM<space>dd**.

The details of the musicians are stored in the first seven lines of a text file called **MusicData.txt.** A line of the file representing the data of a musician has the following format:

<name>;<stageName>;<genre>;<start>

For example the first line is:

```
Jenny Sampson;Rebel;4;2023 03 13
```

Jenny Sampson has the stage name Rebel, a genre of 4 and a start date of 2023 03 13.

The second line is:

```
Thabisile Ndlovu;;1;2022 09 26
```

Thabisile Ndlovu does not have a stage name, so that field is blank, a genre of 1 and a start date of 2022 09 26.

The first 7 lines of the text file are shown below:

```
Jenny Sampson;Rebel;4;2023 03 13
Thabisile Ndlovu;;1;2022 09 26
Suse Mai;Flowers;1;2012 11 27
Samuel Ericson;;1;2019 10 19
Kabelo Mthembu;Easy K;2;2011 07 26
Kusuma Chapman;;3;2021 06 12
Roger Hatfield;;3;2024 02 01
```

Song Class

This class will represent the basic details of songs.

- **artistName** – the full name of the musician who created the song, stored as a string.
- **title** – the title of the song, stored as a string.
- **likes** – the number of likes the song has, stored as a number.
- **dislikes** – the number of dislikes the song has, stored as a number.

The details of the songs are stored after the seven lines of musician data in the file **MusicData.txt.** A line of the file representing the data of a song has the following format:

<artistName>;<title>;<likes>;<dislikes>

For example, the first line is:

```
Samuel Ericson;Running Energy;7355;6987
```

The song is called Running Energy, sung by Samuel Ericson. It has 7355 likes and 6987 dislikes.

The first 10 lines of Song data from the text file are shown below:

```
Samuel Ericson;Running Energy;7355;6987
Kusuma Chapman;Highway Wilderness;3113;841
Jenny Sampson;Friday Cocktail;6127;4657
Kusuma Chapman;Lighter Lola;9238;7206
Samuel Ericson;No Chance;4986;4737
Jenny Sampson;Discover Light;7855;0
Kabelo Mthembu;Babona;9439;8118
Suse Mai;Change The Fusion;1018;601
Kabelo Mthembu;Walking With Gqom;8141;6187
Thabisile Ndlovu;First Heart;7979;11011
```

NOTE: The file **MusicData.txt** contains more songs than those shown above.

QUESTION 2

Use the class diagram below to create a new class called **Song**. This class will be used to store the details of a song created by a musician. The diagram below indicates the fields and methods that are required.

Song
Fields: - artistName : string - title : string - likes : integer - dislikes : integer
Methods: + Constructor(inAN : string, inTitle : string, inL : integer, inD : integer) + getArtistName(): string + getTitle() : string + getLikes() : integer + getWellLiked() : boolean + toString() : string

2.1 Create a new class named **Song** with the **artistName**, **title**, **likes** and **dislikes** fields, as shown in the class diagram. These fields must not be accessible outside the class. (3)

2.2 Write code to create a parameterised constructor method that will accept parameters for all fields, as shown in the class diagram and assign these values to the corresponding fields. (3)

2.3 Create accessor methods for the **artistName**, **title** and **likes** fields. (2)

2.4 A song is well liked if the number of likes is greater than the number of dislikes. Code a method called **getWellLiked**. This method should return whether the song is well liked or not. (3)

2.5 Create a **toString** method that will return a string that combines the values of the **title**, **likes** and **dislikes** fields in the **Song** class. The format of the string should be as follows:

title<tab>**likes**<tab>**dislikes**

For example:

```
Running Energy   7355     6987
```

(2)
[13]

QUESTION 3

Use the class diagram below to create a new class called **Musician**. This class will be used to store the details of a musician and the genre of their music.

Musician
Fields: - name : string - stageName : string - genre : integer - start : date + ROCK = 1 : integer + GQOM = 2 : integer + AMAPIANO = 3 : integer + TECHNO = 4 : integer
Methods: + Constructor(inName : string, inStage : string, inGenre : integer, inStart : date) + getName() : string + getStart() : date + getGenre() : string + setGenre(inGenre : integer) + toString() : string

3.1 Create a new class named **Musician** with the **name**, **stageName**, **genre** and **start** fields as shown in the class diagram. These fields must not be accessible outside the class. (3)

3.2 Add the class constants **ROCK**, **GQOM**, **AMAPIANO** and **TECHNO** as shown in the class diagram. (2)

3.3 Code a parameterised constructor method that will accept parameters for the name, stage name, genre and start date fields as shown in the class diagram. Assign these values to the **name**, **genre** and **start** fields of the class.
Assign the **stageName** field as follows:
- If the **inStage** parameter is an empty string, then assign **inName** to the **stageName** field.
- Otherwise, assign the **inStage** parameter to the **stageName** field. (5)

3.4 Add accessor methods for the **name** and **start** fields of the class. (2)

3.5 Code a **getGenre** method to return a string representing the genre of the **Musician**. This method must return:

- '**Rock**' if the genre is **ROCK**
- '**Gqom**' if the genre is **GQOM**
- '**Amapiano**' if the genre is **AMAPIANO**
- '**Techno**' if the genre is **TECHNO**
- '**Uncategorised**' if the genre is anything else (4)

3.6 Code a **setGenre** method as shown in the class diagram to assign the genre of the **Musician**. (2)

3.7 Code a **toString** method to return the **stageName**, **genre** and **start** fields on the same line. The genre must be displayed as a text value and not as an integer.

The format should be as follows:
stageName<tab>**genre**<tab>**start**

For example:
```
Rebel       Techno       2023-03-13
```

(2)
[20]

QUESTION 4

4.1 Create a class called **MusicManager**. (1)

4.2 Create four fields for the **MusicManager** class as described below:
An array called **musArr** to store up to 100 **Musician** objects.
A counter called **musSize** to count the objects added to **musArr**.
An array called **songArr** to store up to 100 **Song** objects.
A counter called **songSize** to count the objects added to **songArr**.
The fields should not be accessible outside the class. (3)

4.3 Create a constructor method that will read the contents of the text file **MusicData.txt** containing the information about **Musicians** and **Songs**. The first seven lines contain data about **Musicians**. You may hardcode the number 7 in your solution. The remaining lines contain data about **Songs**. The method should do the following:

- Check if the file **MusicData.txt** exists.
- Display a suitable error message if the file does not exist.
- Open the file for reading.
- Loop through the first seven lines of the text file. In each iteration of the loop:
  - Read the line and split the data into separate parts.
  - Convert the contract start data into a Date object (remember that the data is stored in the text file in the format **yyyy<space>MM<space>dd**).
  - Create a **Musician** object and store the object in the next available position in the **musArr** array.
  - Update the counter variable **musSize**.
- Loop through the remaining lines. In each iteration of the loop:
  - Read the line and split the data into separate parts.
  - Create a **Song** object and store the object in the next available position in the **songArr** array.
  - Update the counter variable **songSize**. (15)

4.4 Write code to create a **toString** method. This method should return a string with:

- The heading 'Musicians'.
- Underneath the heading the details of all **Musicians**, each on a new line.
- A blank line.
- The heading 'Songs'.
- Underneath the heading the details of all **Songs**, each on a new line as shown in Question 5.3.

Use the **toString** methods created in the previous classes. (6)

4.5 Code a method named **getLongTermMusicians**. This method should return the name and length of time since the **Musician** started their contract only for **Musicians** who started their contract more than 5 years from the current date. Do NOT hardcode the current date.

The format must be as follows:
name<space>**numYears**<space>"years."

For example:
```
Suse Mai 11 years.
```
(6)

[31]

QUESTION 5

5.1 Write code to create a text-based user interface called **MusicUI** that will allow simple input and output. (1)

5.2 Declare and instantiate a **MusicManager** object. (1)

5.3 Write code to call the appropriate method in the **MusicManager** class to display a list of all the **Musician** objects and the **Songs** they have written. The sample output is shown below. Note that all musicians are displayed, but only the first 10 songs are shown in the sample output. Your code should list **all** songs:

```
Musicians
Rebel                       Techno          2023-03-13
Thabisile Ndlovu            Rock            2022-09-26
Flowers                     Rock            2012-11-27
Samuel Ericson              Rock            2019-10-19
Easy K                      Gqom            2011-07-26
Kusuma Chapman              Amapiano        2021-06-12
Roger Hatfield              Amapiano        2024-02-01

Songs
Running Energy              7355            6987
Highway Wilderness          3113            841
Friday Cocktail             6127            4657
Lighter Lola                9238            7206
No Chance                   4986            4737
Discover Light              7855            0
Babona                      9439            8118
Change The Fusion           1018            601
Walking With Gqom           8141            6187
First Heart                 7979            11011
```

(1)

5.4 Write code to call the appropriate method in the **MusicManager** class to display all long term **Musicians**. The correct output is given below:

```
Suse Mai 11 years.
Kabelo Mthembu 12 years.
```

(1)
[4]

QUESTION 6

6.1 In the **MusicManager** class, code a method called **adjustGenre** to change the genre for a specified musician. This method should accept the name of a **Musician** and a genre as parameters. The method should set the genre of the correct **Musician** to the new genre. The method should not return anything. (6)

6.2 In the **MusicUI** class, call the **adjustGenre** method. Send 'Samuel Ericson' as the name parameter and TECHNO as the integer parameter. (2)

6.3 In the **MusicManager** class, code a helper method called **countSongs**. This method should accept the name of a **Musician** as a parameter. The method should return the number of **Song** objects whose **artistName** matches the parameter. (6)

[14]

QUESTION 7

7.1 Code a method called **listEasyListeners**.

- An easy listener is any song which has more than 5000 likes and is well liked (has more likes than dislikes).
- The method must create a string with the name and easy listening songs from each **Musician**.
- The first line should be the **Musician's** name. Underneath that line should be each easy listening song from that **Musician**.
- There should be a blank line separating each musician's easy listening songs from the next musician.
- If a **Musician** has recorded songs, but none of them are easy listening songs then add 'No suitable songs' instead of the list of easy listening songs to the string.
- If the **Musician** has not recorded any songs add 'No songs recorded' instead of the list of easy listening songs to the string. (17)

7.2 Call the **listEasyListeners** method in the appropriate place and display the string. (1)

The correct output is shown below:

```
Jenny Sampson Easy Listeners
Friday Cocktail
Discover Light
Fragments of Interest
Sick Madness

Thabisile Ndlovu Easy Listeners
No suitable songs

Suse Mai Easy Listeners
Absolute Train
Soft Killer

Samuel Ericson Easy Listeners
Fearless Eyes

Kabelo Mthembu Easy Listeners
Babona
Walking With Gqom
Morning Starring
Feel The Moves
Crazy Masters

Kusuma Chapman Easy Listeners
Lighter Lola

Roger Hatfield Easy Listeners
No songs recorded
```

[18]

100 marks

Total: 150 marks