NATIONAL SENIOR CERTIFICATE EXAMINATION
NOVEMBER 2016

INFORMATION TECHNOLOGY: PAPER II

Time: 3 hours 120 marks

PLEASE READ THE FOLLOWING INSTRUCTIONS CAREFULLY

1. This question paper consists of 14 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).

5. Make sure 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 in each question. For example, if the question does not ask for data validation, then no marks are awarded for it, and therefore no code needs to be written.

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

8. When accessing files from within your code, DO NOT use full path names of the file, as this will create problems when the program is marked on a computer other than the one you are writing on. Merely refer to the files using their names and extensions, where necessary.

9. 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. You are advised to look at the supplied data files carefully.

10. Make sure that routines such as searches, sorts and selections 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.

11. All data structures must be defined and declared by you, the programmer. You may not use components provided within the interface to store and later retrieve data.

12. 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 which will be more efficient considering the questions that are asked in the paper.

13. You must save all your work regularly on the disk you have been given, or the disk space allocated to you, for this examination.

14. If there is a technical interruption that prevents you from writing your examination, e.g. a power failure, when you resume writing your examinations, you will only be given the time that was remaining when the interruption began. No extra time will be given to catch up on work that was not saved.

15. Make sure that your examination number appears as a comment in every program that you code as well as on every page of hard copy that you hand in.

16. Print a code listing of all the programs/classes 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 for the printing of your work.

SCENARIO

TieTheKnot is a company that specialises in developing software solutions specifically for weddings. They consult to a variety of companies like department stores, wedding photographers and wedding venues in order to develop simple and effective software solutions based on a wedding theme. You are required to help them develop a database for a gift registry and a programme that manages wedding bookings for a wedding venue.

SECTION A STRUCTURED QUERY LANGUAGE

TieTheKnot wants to keep track of the wedding gift registries that couples have set up. Couples that are getting married often set up a list of gifts (called a gift registry) they would like to receive in order to aid their wedding guests in purchasing gifts. The couple makes a list of all the gifts they would like to receive (this is the gift registry) and wedding guests can then go to the retailer that sells that item, check the list for that couple and purchase a gift off the list. Many retailers will list their items on the registry. The system then keeps track of which gifts have been purchased for which couples so that they do not receive duplicates. Couples may want more than one of a particular item (e.g. 12 wine glasses) and the registry allows them to indicate this. The registry then keeps track of how many of that item has been purchased for a particular couple, from a particular retailer. This allows more than one wedding guest to purchase the same gift until the required quantity is reached.

The database **WeddingDB** contains three tables. The first table **tblItems** contains the details of each item that is available as a gift including their ItemID, description, price and retailer. One item can appear on many individual couple's registries. The second table **tblCouples** contains information on each of the couples that have signed up for the registry including the CoupleID, the name of the bride, the name of the groom, and their wedding date. The third table **tblGifts** contains details pertaining to which couples have asked for which items, how many of each item they have requested, how many of each item have been purchased and whether there are any more available to purchase. Even though the available field can be calculated, it has been explicitly included for the purpose of this question. You are required to extract some useful information from the database.

The fields in the database are discussed below. Below each description is a screenshot of the first 10 rows of data for your convenience. The tables do contain more data:

tblItems

Field Name	Data Type	Description
ItemID	Number	This field assigns a unique ID for each item as a number.
Description	Text	This field contains a description of the item as text.
Price	Currency	This field contains the price of the item as a currency in rand.
Retailer	Text	This field contains the name of the retailer where the item is available as text.

tblItems			
ItemID	**Description**	**Price**	**Retailer**
12337	Seafood Aluminium:Crab & Lobster	R604.95	Decor Centre
12446	Tray Initial Set	R274.95	Home Pride
13121	Plate Rack	R219.95	Furnmart
13738	Server Set:Classic	R164.95	Home Pride
13941	Crawfish Tumbler	R120.95	Mr House and Home
14087	Ladle:Acrylic	R65.95	Mr House and Home
14195	Bowl:Square 3 in 1	R604.95	Decor Centre
15570	Beaded Aluminium:Bowl Oval	R549.95	Furnmart
15977	Bowl:Pasta	R329.95	Mr House and Home
17052	Beaded Aluminium:Platter Square	R472.95	Decor Centre

tblCouples

Field Name	Data Type	Description
CoupleID	AutoNumber	This field assigns a unique ID for each couple. This field is an autonumber field.
BrideName	Text	This field contains the name of the bride as text.
GroomName	Text	This field contains the name of the groom as text.
WeddingDate	Date/Time	This field contains the date of the couple's wedding as a date/time.

tblCouples			
CoupleID	**BrideName**	**GroomName**	**WeddingDate**
1	Emily Ryan	Harold Grant	2015-12-03
2	Linda Baker	Eric Cook	2015-11-08
3	Kathy Hawkins	Joshua Stevens	2016-08-31
4	Lori Brown	Carlos Diaz	2016-10-20
5	Diane Kennedy	Andrew Woods	2015-12-30
6	Marie Adams	Timothy Adams	2015-12-17
7	Sarah Carroll	Stephen Lawson	2015-12-15
8	Wanda Ferguson	Albert Richardson	2016-04-13
9	Susan Smith	Ryan Jones	2016-07-30

tblGifts

Field Name	Data Type	Description
ItemID	Number	This field contains the ID of the item requested by a couple for their gift registry. This field is a foreign key to tblItems.
CoupleID	Number	This field contains the ID of the couple that requested the item. This field is a foreign key to tblCouples.
RequestedQuantity	Number	This field contains the quantity of the item requested by the couple as a number.
BoughtQuantity	Number	This field contains the number of items already purchased for a couple as a number.
Available	Boolean	This field contains whether there are still more items available for purchase for that couple as a Boolean. The values in this field are all intentionally set to False.

tblGifts				
ItemID	**CoupleID**	**RequestedQuantity**	**BoughtQuantity**	**Available**
12337	2	2	1	False
12337	4	1	1	False
12446	6	1	1	False
13738	3	1	0	False
13941	5	2	0	False
13941	8	4	4	False
14087	6	1	1	False
14087	7	4	1	False
14195	5	4	4	False
14195	6	2	1	False

1.1 Write a query that will list all the details of all items from **tblItems** that are available from the retailer "Home Pride". (3)

1.2 Write a query that will list the description and price of all items from **tblItems** that cost between R350 and R500 (inclusive of both values). (4)

1.3 Write a query that will list all the couples in **tblCouples**. You must list the couple name as well as the month of their wedding and the results should be ordered by month in ascending order. A new field for the couple name should be called **CoupleName** and consist of the **BrideName**, followed by an **&** symbol, followed by the **GroomName**. The correct output is given below:

CoupleName	MonthOfWedding
Wanda Ferguson & Albert Richardson	4
Susan Smith & Ryan Jones	7
Kathy Hawkins & Joshua Stevens	8
Lori Brown & Carlos Diaz	10
Linda Baker & Eric Cook	11
Sarah Carroll & Stephen Lawson	12
Marie Adams & Timothy Adams	12
Diane Kennedy & Andrew Woods	12
Emily Ryan & Harold Grant	12

(4)

1.4 Currently the **Available** field for each record in **tblGifts** is set to false. This field should be **true** for any requested gift which has a **BoughtQuantity** that is less than the **RequestedQuantity**. Write a query that will update the **Available** field in the **tblGifts** table accordingly. (4)

1.5 Write a query that will show the average price of all items containing the word "platter" for each retailer in **tblItems**. The output should be as follows: (5)

Retailer	AvgItemPrice
Decor Centre	R511.45
Furnmart	R274.95
Home Pride	R1,759.95

1.6 Write a query that will display the total **Price** of all items not listed in any couple's registry. Your output should be as follows: (5)

TotalPrice
R5,609.15

1.7 Write a query that will display which gifts (and how many) are available for each couple. You should display the **BrideName, Description** of the item and new field called **QuantityAvailable** in your output. **QuantityAvailable** is the number of items still available for that couple for that particular gift: use the **RequestedQuantity** and **BoughtQuantity** fields to calculate this value. Order your output by **BrideName** alphabetically and then by the quantity available from smallest to largest. The order of the **BrideName** and **QuantityAvailable** columns is important in your output; the **Description** column order may vary. The first 12 rows of the correct output are given below: (9)

BrideName	Description	QuantityAvailable
Diane Kennedy	Monkey Bread Cooker	1
Diane Kennedy	Seafood Aluminium:Crab Bowl	1
Diane Kennedy	Pitcher:Acrylic	1
Diane Kennedy	Square Crab Glass	1
Diane Kennedy	Vase Set	1
Diane Kennedy	Platter:FDL Aluminium	2
Diane Kennedy	Crawfish Tumbler	2
Diane Kennedy	Vase:Verde Round	2
Diane Kennedy	Tile:Wedding Damask	3
Emily Ryan	Fleur De Lis Pillow	1
Emily Ryan	Crab Tumbler	1

1.8 Susan Smith and Ryan Jones (**CoupleID 9**) do not have any gifts registered in **tblGifts**. They decide they want all the same gifts (and the same quantities) as Linda Baker and Eric Cook **(CoupleID 2**). Write a query to insert the same gifts that **CoupleID 2** has requested into the **tblGifts** for **CoupleID 9**. The new records should have a **BoughtQuantity** of 0 and an **Available** of **True**.

You must write a single SQL statement to generate the records for **CoupleID 9**. (6)

40 marks

SECTION B OBJECT-ORIENTED PROGRAMMING

TieTheKnot has been contracted by a popular wine estate in Cape Town that wants to start offering their beautiful grounds as a venue for weddings. The wine estate already has 50 bookings for November 2016 and needs some software to help them process these bookings.

The wine farm has two venues that are suitable for weddings:
- Garden venue – this venue can hold up to 130 guests for a single wedding.
- Lake venue – this venue can hold up to 160 guests for a single wedding.
- There is only one wedding per venue on any given date.

Each **Wedding** has the following information:
- Bride – the full name of the bride.
- Groom – the full name of the groom.
- Wedding Date – the date of the wedding in November 2016.
- Venue – the booked venue (either "Garden" or "Lake").
- Number of Guests – the total number of guests attending the wedding.

The wine farm wants to know which weddings are booked for a particular day in November 2016 so they want to store the following information for each booking day in the month.

Each **BookingDay** contains the following information:
- Date – the full date of each day in the month. Possible values are any date from 2016-11-01 to 2016-11-30.
- Garden – the details of the wedding booked for the Garden venue, if any.
- Lake – the details of the wedding booked for the Lake venue, if any.

There will be some days in November where both venues have been booked, some days where only one of the two has been booked and some days where neither venue has been booked.

You have been given a file called **weddings.txt** which contains the information of exactly 50 weddings that have already been booked with the venue for November 2016. There will always be exactly 50 weddings in the given file. The weddings are not listed in any particular order and the file does not contain more than one wedding per venue for a particular date. A sample of the first 10 lines of the file is given below:

```
Chiquita Sanford#Magee Sosa#2016-11-05#Garden#84
Sade Knox#Logan Duncan#2016-11-30#Garden#101
Kirestin Tyler#Kirk Palmer#2016-11-03#Lake#146
Ayanna Rivas#Abel Hopkins#2016-11-30#Garden#139
Hadassah Mason#Carlos Hoffman#2016-11-28#Garden#120
Jessamine Max#Malcolm Cass#2016-11-06#Garden#125
Kevyn Mcintosh#Kermit Mick#2016-11-12#Garden#95
Celeste Chandler#Dieter Chaps#2016-11-10#Garden#99
Lacota Buchanan#Devin Bass#2016-11-29#Garden#141
Nevada Summers#Brenden Sot#2016-11-12#Lake#147
```

Each line of this file contains the information of a single wedding booking for a particular venue in the month of November. The line is formatted as follows:

```
bride#groom#weddingDate#venue#guests
```

QUESTION 2

Use the class diagram below to create a new class called **Wedding**. This class will be used to create objects that will store the details of one wedding booking. The diagram below indicates the properties and methods that are required.

Take careful note of the method and parameter names in the UML diagram.

Wedding
Properties: – String bride – String groom – String weddingDate – String venue – Integer guests
Methods: + Constructor(String inBride, String inGroom, String inDate, String inVenue, Integer inGuests) + getWeddingDate() : String + getVenue() : String + getGuests() : Integer + equals(String inDay, Integer inVenue) : Boolean + toString() : String

2.1 Write code to create a new class called **Wedding**. (1)

2.2 Write code to create the five properties for the **Wedding** class as indicated in the above class diagram. (3)

2.3 Write code to create a constructor method that accepts four strings and one integer as parameters that represent the bride's name, groom's name, wedding date, venue and number of guests respectively. These parameters should be used to set the values of the five properties of the **Wedding** class. (3)

2.4 Write code to create accessor methods for the **weddingDate**, **venue** and **guests** properties of the **Wedding** class. Each of these three methods should be named appropriately and return the correct type and value for each property respectively. (3)

2.5 Write code to create a method called **equals** that returns a Boolean. This method should accept two strings as parameters which indicate a day and venue respectively. This method should return true if the wedding's date and venue properties match the date and venue passed as parameters, and false otherwise. (4)

2.6 Write code to create a **toString** method that will return a string representing the wedding's information. The format is as follows:

```
bride & groom<tab><tab>guests<space>"guests"
```

for example:

```
Chiquita Sanford & Magee Sosa       84 guests
```

(3)

[17]

QUESTION 3

Use the class diagram below to create a new class called **BookingDay**. This class will be used to store the details of the bookings for a single day in a month. Two of the properties are **Wedding** objects (the class created in the previous question). The diagram below indicates the properties and methods that are required.

BookingDay
Properties: - String bookingDate - **Wedding** garden - **Wedding** lake
Methods: + Constructor(String inBDate, Wedding inGarden, Wedding inLake) + getOverbooked : String + toString() : String

3.1 Write code to create a new class called **BookingDay**. (1)

3.2 Write code to create three properties that will store the **bookingDate, garden** and **lake** weddings associated with a **BookingDay**. Choose appropriate data types for these properties but note that the **garden** and **lake** properties are objects of the class **Wedding**. These properties should not be visible from outside the class. (3)

3.3 Write code to create a constructor method that will initialise all the properties of the **BookingDay** class. Note that in addition to the bookingDate parameter the second and third parameters are objects of the **Wedding** class. Use the parameters to initialise the properties of the **BookingDay** class. (4)

3.4 Write code to create a method called **getOverBooked** that should return a string indicating that venues (if any) are overbooked on that date. A venue is overbooked if the number of guests exceeds the capacity of the venue. The garden venue can hold 130 guests and the lake venue can hold 160 guests. Use the **getGuests** method of the **garden** and **lake** attributes to determine the number of guests for each venue on that day and return a string containing information on which venue is overbooked and by how many guests. As an example your output for a date where the garden venue is booked for 144 guests and the lake venue is booked for 167 guests should be as follows:

```
Garden venue is overbooked by 14 guests<newline>
Lake venue is overbooked by 7 guests<newline>
```

(5)

3.5 Write code to create a **toString** method which will return a string comprised of the information for the booking day. The string returned should be in the following format:

```
Booking Date:<newline>
Garden : <garden wedding details><newline>
Lake   : <lake wedding details><newline>
<overbooking information>
```

Use the toString methods of the **garden** and **lake** respectively to display the information for the garden and lake wedding. Note that there may not be one or both of those weddings on a particular day. Use the **getOverBooked** method to display the overbooking information. For example:

```
2016-08-14
Garden : Vera Hatfield & Yuli Schwartz     144 guests
Lake   : Chanda Olsen & Byron Everett      124 guests
Garden venue is overbooked by 14 guests
```

Note that there may not be a garden and/or lake wedding booked (then garden and/or lake **Wedding** objects may be **null**/**nil**) for that day and thus the **garden** and/or **lake** attributes may not have values. If this is the case return the following string:

```
2016-11-15<newline>
No weddings booked
```

(5)
[18]

QUESTION 4

4.1 Write code to create a new class called **VenueManager**. (1)

4.2 Write code to declare the following two arrays as instance variables:
- An array that can be used to store exactly 50 **Wedding** objects.
- An array that can be used to store exactly 30 **BookingDay** objects (one object for each day in the month of November 2016). (4)

4.3 Write code to create a constructor method that will read the contents of the file **weddings.txt**. The file name should be given as a String parameter to the constructor method. Each line will result in one **Wedding** object being added to the array. Do the following:
- Check if the file exists. If it does not, display an error message.
- Open the file for reading.
- Loop through the file 50 times. In each iteration of the loop:
  - Read in each line and split the **Wedding** data contained in that line into the names of the bride and groom, wedding date, venue and number of guests.
  - Create a **Wedding** object using the data.
  - Add the **Wedding** object to the array.
  - Note that you do not need to create any **BookingDay** objects in this method. (10)

4.4 Write code that will create a method called **listAllWeddings**. This method should return a string that contains the information of all weddings. Use the object's **toString** methods that you created in the questions above. Each wedding's details should be on a new line. (5)

4.5 Write code to create a method called **getWeddingsOnDay** that will search the array of wedding objects for all weddings on a particular day and return a **Wedding** object. This method should accept two string parameters: the first represents the date to search for and the second parameter the venue to search for. The method must return the wedding object that occurs on that day and in that venue specified by the method's parameters. If no match is found, return **null/nil**. There will only ever be at most one wedding per venue on a given date. Any search is acceptable. (5)

4.6 Write code to create a method called **processBookings** that will populate the bookings array with **BookingDay** objects and return a string with the information for each booking. Do the following for each booking:
- Generate a date corresponding to each day in November.
- Ascertain the garden and lake weddings booked for that day (if any).
- Create a new **BookingDay** object using the correct parameters and add the object to the array.
- Return a string containing the information for each **BookingDay** object on a new line. (8)

[33]

QUESTION 5

5.1 Write code to create a simple user interface called **WeddingUI** that will allow simple output. (1)

5.2 Declare and instantiate a **VenueManager** object at the appropriate place in the code. Call the constructor using "**weddings.txt**" as the filename argument. (1)

5.3 Write code that will display the following by calling the appropriate methods in the **VenueManager** class. You must call the methods in the following order:

List all Weddings
Populate and Display Booking Dates (2)
[4]

QUESTION 6

You are now required to write code to display a simple monthly calendar for a month with 30 days. The calendar will have a column for each day of the week and the rows will indicate the individual weeks and which days fall on which day of the week. For example:

M	Tu	W	Th	F	Sa	Su
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

Note that the calendar will vary depending on which day of the week the 1st of the month falls on. In the example above the 1st of the month is on a Thursday and the days of the week progress accordingly. Your code should be able to accept any starting day (1 = Monday, 2 = Tuesday, 3 = Wednesday, 4 = Thursday, 5 = Friday, 6 = Saturday and 7 = Sunday) and display the calendar accordingly.

6.1 Write code to add a method in the **VenueManager** class called **displayCalendar**. This method should accept an integer parameter indicating the day of the week the 1st falls on. E.g. if the first day of the month is a Friday the parameter will have the value of 5. The method should then return a string representing the calendar for the month. **Note that you may not make use of any built-in Calendar or Date classes, libraries or methods when answering this question.** (7)

6.2 Write code in **WeddingUI** to call the **displayCalendar** method and display the calendar for a 30-day month starting on a particular day. If you pass 7 (representing a start day of Sunday) as the integer parameter then your output should be as follows:

```
 M    Tu    W    Th    F    Sa    Su
                                   1
 2    3     4    5     6    7     8
 9    10    11   12    13   14    15
16    17    18   19    20   21    22
23    24    25   26    27   28    29
30
```

(1)
[8]

80 marks

Total: 120 marks