NATIONAL SENIOR CERTIFICATE EXAMINATION
NOVEMBER 2013

INFORMATION TECHNOLOGY: PAPER II

Time: 3 hours 120 marks

PLEASE READ THE FOLLOWING INSTRUCTIONS CAREFULLY

1. This question paper consists of 12 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 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 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 the 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

The National Theatre School of South Africa is hosting a William Shakespeare Festival in November and wants to use various computer systems when organising this event. You are required to assist them with various database and programming related work.

SECTION A STRUCTURED QUERY LANGUAGE

You are required to extract some useful information from the database of shows and actors for the purpose of the festival.

There are three tables in the database to store information about the shows (all Shakespearean plays) and the actors performing in these productions. The actors' personal details are stored in a table named **tblActors**, the information about the various shows (for example, dates, time, capacity and cost) are stored in a table named **tblShows** and a list of which actors are playing the roles in each play are listed in a table named **tblRoles**.

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

tblActors

ActorID	This is an automatic numbering field which assigns each actor a unique ID number
ActorName	This field contains the names of the actors
ActorDOB	This field contains the dates of birth of the actors
ActorRate	This field contains the amount of money in Rands that the actor earns each hour or part thereof

ActorID	ActorName	ActorDOB	ActorRate
1	Garth York	1980/09/05	R 450.00
2	Ebony Sweeney	1981/02/23	R 300.00
3	Mary Rocha	1962/01/31	R 750.00
4	Summer Dickson	1970/07/30	R 275.00
5	Clarke Stafford	1969/03/17	R 375.00
6	Veronica Decker	1983/01/10	R 150.00
7	Jason Hyde	1963/10/15	R 225.00
8	Odette Barker	1982/07/31	R 225.00
9	Oprah Buckner	1982/12/11	R 550.00
10	Amir Collier	1965/11/09	R 750.00

tblShows

ShowID	This field contains a unique ID for each of the shows
ShowName	This field contains the name of the play
ShowDate	This field contains the date that the show is being performed on
ShowTime	This field contains the time the show is scheduled to start
ShowDuration	This field contains the duration of the show in minutes
ShowCapacity	This field contains the maximum number of people who can attend this particular show
ShowCost	This field contains the cost of a single ticket to the show

ShowID	ShowName	ShowDate	ShowTime	ShowDuration	ShowCapacity	ShowCost
12Night	Twelfth Night	2013/11/08	15:00	160	325	200
2Gents	Two Gentlemen of Verona	2013/11/20	15:00	215	350	200
Ado	Much Ado about Nothing	2013/11/03	15:00	109	415	200
Errors	Comedy of Errors	2013/11/24	20:00	155	300	350
Henry5	Henry V	2013/11/12	17:30	270	300	250
Lear	King Lear	2013/11/04	17:30	179	350	250
Macbeth	Macbeth	2013/11/09	17:30	174	280	250
Merchant	Merchant of Venice	2013/11/20	17:30	279	400	250
MND	Midsummer Night's Dream	2013/11/24	15:00	204	375	200
Othello	Othello	2013/11/20	20:00	190	300	350

tblRoles

ActorID	This field contains the ID of the actor that is playing the role. This key is a foreign key for the tblActors table
RoleName	This field contains the name of the role or character being played by the actor
ShowID	This field contains the ID of the show the role or character is associated with. This key is a foreign key for the tblShows table

ActorID	RoleName	ShowID
1	Olivia	12Night
1	Third Outlaw	2Gents
1	Don Pedro	Ado
1	Aemilia	Errors
1	Duke of Bourbon	Henry5
1	Banquo	Macbeth
1	Lorenzo	Merchant
1	Hippolyta	MND
1	Sailor	Othello
2	Hero	Ado

QUESTION 1

1.1 Write a query that will list all the details of all the shows from **tblShows** in date order. (3)

1.2 Write a query that will list the **ShowName**, **ShowTime** and **ShowCapacity** of all shows that start at 15:00 and have a capacity of 250 people or more. (3)

1.3 Write a query that will display the age of each actor in years. Display the **ActorName** as well as the age in a field called **ActorAge**. (3)

1.4 Write a query which will list how many roles there are for each of the shows. You should display the **ShowName** and the total number of roles. (5)

1.5 Write a query that will add a new actor, with the name of Bob Reeves born 4 October 1976 to **tblActors**. Bob Reeves is paid a rate of R175 per hour. (4)

1.6 The capacity of the show 'Henry V' with **ShowID** 'Henry5' on the 2013-11-12 has been changed to 450. Write an update query that will update this record in **tblShows**. (2)

1.7 Write a query which will calculate the total income in ticket sales that the theatre will receive each day if they fill all their shows to their capacity. The total income for a show can be calculated as follows:

*Total Income = ShowCapacity * ShowCost*

Show the date as well as the total income for all shows on each date. (4)

1.8 Write a query which will list all roles which begin with 'Sir'. You must list the **ShowName** and **RoleName** of each role. An example of the output is given below: (6)

ShowName	RoleName
Twelfth Night	Sir Andrew Agueche
Henry V	Sir Thomas Erpingha
Henry V	Sir Thomas Grey
Twelfth Night	Sir Toby Belch

1.9 Write a query that will display how much each actor will earn for each play they are in during the course of the festival. Display the **ActorName**, **ShowName** and total amount to be paid to that actor (round this field to two decimal places). Actors get their hourly rate for the entire duration of the play. An example of the output is given below: (10)

ActorName	ShowName	TotalEarnings
Amir Collier	Henry V	3375
Amir Collier	King Lear	2237.5
Amir Collier	Macbeth	2175
Amir Collier	Merchant of Venice	3487.5
Amir Collier	Midsummer Night's Dream	2550
Amir Collier	Romeo and Juliet	2000
Amir Collier	Two Gentlemen of Verona	2687.5
Brittani Puckett	Comedy of Errors	1937.5
Brittani Puckett	Henry V	3375
Brittani Puckett	King Lear	2237.5
Brittani Puckett	Macbeth	2175
Brittani Puckett	Midsummer Night's Dream	2550
Brittani Puckett	Othello	2375
Brittani Puckett	Romeo and Juliet	2000
Brittani Puckett	Twelfth Night	2000
Brittani Puckett	Two Gentlemen of Verona	2687.5
Cade Ewing	Henry V	1237.5
Cade Ewing	King Lear	820.42

40 marks

SECTION B OBJECT ORIENTED PROGRAMMING

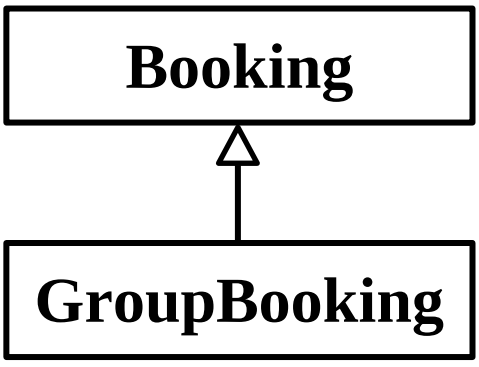
The theatre at the National Theatre School of South Africa holds 180 people and the organisation keeps track of the bookings for each show separately. The theatre layout has 12 rows of chairs with 15 chairs in each row. Rows are labelled from A (at the front) to L (at the back).

The theatre allows theatre-goers to book for shows via an online website called TechnoTickets. The bookings are written to a text file called **bookings.txt**. This file needs to be processed to determine booked seats and outstanding seats. For any given show bookings can either be **individual bookings** or **group bookings**.

Individual bookings are single bookings with a seat reference, contact number and payment status recorded. Each booking is made separately and is not connected to any other booking.

Group bookings are when 4 or more consecutive seats are booked. In addition to seat reference, contact number and payment status, a group name and group size is recorded for each group booking. For example, a group booking of 10 seats will be 10 separate bookings linked by the group name and size.

After some consideration you decide on the following class structure:



You have been given a file called **bookings.txt** which contains a list of bookings for a single show at the theatre. The first 25 lines of the file are provided below for your reference (the actual file contains many more lines and you are advised to study it closely).

```
A1,074-061-3602,Yes
A2,074-625-3484,Yes
A3,083-219-2489,Yes
A5,073-345-5087,Yes
A6,082-147-4323,Yes
A7,084-410-8555,No
A8,073-984-5555,Yes
A9,082-122-4323,Yes
A10,084-466-8701,No
A11,073-977-5087,Yes
A12,082-147-4553,Yes
A13,073-984-5087,Yes
A14,072-147-4323,Yes
A15,083-410-8701,No
B1,082-604-3857,Yes,Kolby Girls High School,12
B2,082-604-3857,Yes,Kolby Girls High School,12
B3,082-604-3857,Yes,Kolby Girls High School,12
B4,082-604-3857,Yes,Kolby Girls High School,12
B5,082-604-3857,Yes,Kolby Girls High School,12
B6,082-604-3857,Yes,Kolby Girls High School,12
B7,082-604-3857,Yes,Kolby Girls High School,12
B8,082-604-3857,Yes,Kolby Girls High School,12
B9,082-604-3857,Yes,Kolby Girls High School,12
B10,082-604-3857,Yes,Kolby Girls High School,12
B11,082-604-3857,Yes,Kolby Girls High School,12
```


Each line has either three or five fields present and are separated by commas. The first three fields of any line indicate the seat reference that has been booked, the contact phone number for that booking and a 'Yes' or 'No' indicating whether the booking has been paid for in full. Lines in the file that only have three fields indicate an **individual booking**.

Lines which have two extra fields (five fields in total) represent **group bookings**. The additional fields represent the name of the group and the size of the group respectively. If a group of 6 people have placed a booking this will be represented as 6 separate lines in the text file. Each line of the group booking will be identical except for the seat reference which will differ.

In the example data above Kolby Girls High School have booked 12 seats. The seats are B1 to B12. Each line indicates a different seat reference while the other fields are the same. These 12 entries belong to a single group due to the Group Name (the 4th field) being the same (Kolby Girls High School) for each line. The group size of 12 indicates that there are twelve seats booked for this group.

The cost of any booking is based on the seat that is being booked. In addition group bookings get a further discount. The full price of a ticket is R200 but seats in rows further from the front have discounts associated with them. In addition, groups who book get R20 discount for every four members of the group. This means a group of eight will get a further R40 discount on each of their tickets while a group of 15 will get a discount of R60 on each of their tickets. The row discount is applied before the group discount in the case of group bookings.

QUESTION 2

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

Booking
Properties: - String seat - String contact - boolean paid
Methods: + Constructor(String st, String ct, boolean pd) + getCost() : double + getSeat() : String + isPaid() : boolean + toString() : String

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

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

2.3 Write code to create a constructor method that will initialise all three properties of the **Booking** class. (2)

2.4 Write code to create a method called **getCost**. This method should return the cost of the seat based on the row of the seat. The full price of a ticket is R200 and discounts are applied as follows:

Rows A – D : Full Price

Rows E – H : 20% discount

Rows I – L : 40% discount

For example, if the person is seated in row K they should receive 40% discount on their R200 ticket, making the cost of that ticket R200 – R80 = R120. (5)

2.5 Write code to create accessor methods for the **paid** and **seat** properties. (2)

2.6 Write code to create a **toString** method which will return a String comprised of the **seat** reference and the **cost** (rounded to two decimal places). In addition, if the booking has been paid for the letters 'Pd' should be added onto the end of the returned string. The String must be formatted as follows:

```
seat reference <tab> R cost <tab> payment status
```

for example: `A1  R200.00  Pd` (4)

[17]

QUESTION 3

Use the class diagram below to create a new class called **GroupBooking**. This class is a subclass of the **Booking** class and will be used to create objects that will store the details of a group booking. The diagram below indicates the properties and methods that are required.

GroupBooking
Properties: - String groupName - Integer groupSize
Methods: + Constructor(String st, String ct, boolean pd, String gn, Integer gs) + getCost() : double + getGroupName() : String + toString() : String

3.1 Write code to create a new class called **GroupBooking** which extends the **Booking** class. (2)

3.2 Write code to create two properties that will store the **group name** and **group size** associated with a group booking. Choose appropriate data types for these properties. These values should not be visible from outside the class. (3)

3.3 Write code to create a constructor method that will initialise both of these additional properties as well as the properties of the parent class. (3)

3.4 Write code to override the parent class's **getCost** method. The result of this method should be the cost of a standard booking (as per the parent method) with the group discount applied afterwards, for example the Melville Boy Scouts have booked 13 seats in row G as a **group booking**. Seats in row G means they qualify for a 20% discount bringing the cost of each ticket to R160. They receive a further R60 discount on each ticket due to their group size having 3 complete sets of four, bringing the final cost of each of their tickets to R100. (A group receives R20 discount for each set of four people.) (4)

3.5 Write a code to create an accessor method called **getGroupName** which returns the name of the group as a string. (1)

3.6 Write code to override the parent class's **toString** method. In addition to the details returned from the inherited method you should add the group name to the end of the returned String in the following format:

```
seat reference <tab> R cost <tab> payment status
```

for example: `B1  R140.00  Pd  Kolby Girls High School` (3)

[16]

QUESTION 4

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

4.2 Write code to declare two instance variables in the class of a **ONE-DIMENSIONAL** array that can be used to store up to 180 **Booking** or **GroupBooking** objects. Also declare an integer counter to keep track of how many **Bookings** are stored in the array. These two instance variables should not be accessible from outside the class. (4)

4.3 Write code to create a constructor method that will read the contents of the file **bookings.txt**. Each line in the file contains information on a single **Booking** or **GroupBooking** object. Read each line from the file and instantiate the appropriate type of object (**Booking** or **GroupBooking**) and add it into the array. ***Note in the case of GroupBookings you must create an object in the array for each member of the group. This means for a group booking of six you will have six separate GroupBooking objects in the array.*** (8)

4.4 Write code that will create a method called **listAllBookings**. This method should return a string that contains the information of all bookings for the show. Each booking should appear on its own line. Use the object's **toString** methods that you created in the questions above. (5)

4.5 Write code to create a method called **listGroupBookings**. This method should return a String with the name of each group on a new line. Each group name should be listed only once. In other words, if there are six **GroupBooking** objects, you need to list the group name of that group only once. The correct output is as follows: (7)

```
Group Bookings
--------------
Kolby Girls High School
Knysna Retirement Home
Witbank Hockey Club
Mellville Boy Scouts
Cape Town Drama Society
Durban Gentleman's Club
```

4.6 Write code to create a method called **outstandingPayments** to calculate the total amount owing. This method should return a real number that represents the amount of money that is still outstanding. Only objects whose **paid** attribute is false will have their ticket cost added to this total. The correct output is as follows: (5)

```
Outstanding Payments
--------------------
Outstanding Payments : R 8760.0
```

[30]

QUESTION 5

5.1 Write code to create a simple user interface called **TheatreManager** which will declare and instantiate a **BookingManager** object at the appropriate place in the code. (2)

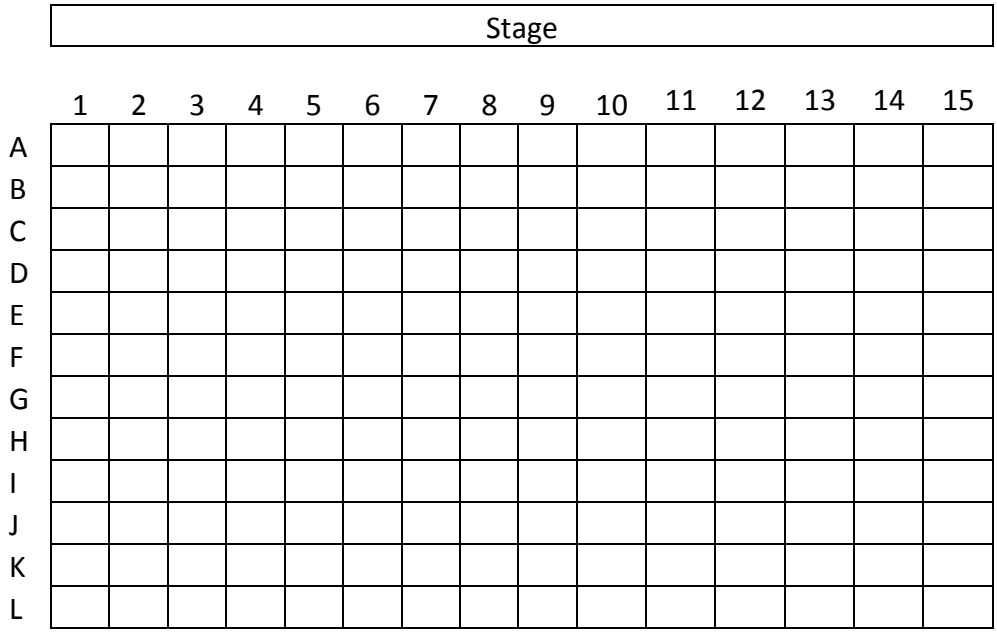
5.2 Write code that will display the following by calling the appropriate methods in the **BookingManager** class. You must call the methods in the following order:

All Bookings
Group Bookings
Outstanding Payments (3)

[5]

QUESTION 6

6.1 The theatre is interested in displaying a list of empty seats based on the bookings they have received and are stored in the array. The following diagram shows the layout of the theatre.



An empty seat is one that is not listed as a seat in the list of bookings. In other words, if the seat reference A4 is not present in any of the bookings it is considered to be an empty seat.

Write code to create the appropriate methods in any of the classes which will help you to achieve this. Your method should return a single string representing the empty seats in the theatre. The correct output is provided below. (11)

```
Empty Seats
-----------
A4 B13 B14 B15 C1 C9 C12 C13 C15 D1 E2 E4 E15 F13 G1 G2 H4 H15 I2 I8 I14 I15 J15
L1 L2 L3 L5 L10 L11 L12 L13 L14 L15
```

6.2 Add a call statement in the **TheatreManager** interface which will display a list of empty seats. (1)

[12]

80 marks

Total: 120 marks