NATIONAL SENIOR CERTIFICATE EXAMINATION
MAY 2024

INFORMATION TECHNOLOGY: PAPER I

Time: 3 hours 150 marks

PLEASE READ THE FOLLOWING INSTRUCTIONS CAREFULLY

1. This question paper consists of 17 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. 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 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 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 for the files. 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 that 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. You should also create a backup of the original files before you start in case the original version is accidentally modified by your solution.

14. 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.

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.

17. You have been provided with 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 provided in MS Access, JavaDB and MySQL format. Ensure that you are able to open the files with the packages that you will use to code your solutions to this examination.

    Section A:
    SummerFun.mdb
    SummerFun_JavaDB.sql
    SummerFun_MySQL.sql
    SQLAnswerSheet.rtf
    SQLBrowser.exe

    Section B:
    clubs.txt

SCENARIO

The company 'Summer Fun' specialises in creating holiday activities for children. Parents can register their children and sign them up for different activities. Some activities run throughout the year and parents may sign their children up for the same activity multiple times during the year. You are an intern at the company working in the database management department. Your manager would like to assess your skill with querying databases and would like you to work with a sample of the database that contains the sign-ups from last year and some from this year.

The database **SummerFun** has been provided for you.

SECTION A STRUCTURED QUERY LANGUAGE

QUESTION 1

tblChildren contains all the details of the children who have been registered.

FIELDS	DATA TYPE	DESCRIPTION
ChildID	INTEGER	A unique autonumber given to each registered child; used as the primary key of the table.
Firstname	TEXT	The first name of the child.
Lastname	TEXT	The last name of the child.
DOB	DATE	The date of birth of the child.
ParentCellphone	TEXT	The number of a parent to contact in case of an emergency.

The first 10 records of tblChildren are shown below:

ChildID	Firstname	Lastname	DOB	ParentCellphone
1	Kaylyn	Furman	2014/06/18	0828842261
2	Pride	Williams	2012/02/25	0627242377
3	Xiluva	Zwane	2012/04/27	0725884662
4	Sibulele	Sibaca	2013/04/01	0625435781
5	Bradley	Ndlangisa	2008/11/01	0739347124
6	Oratile	Hlanti	2007/12/13	0624359939
7	Lyle	Mothwa	2010/05/15	0729324785
8	Ongeziwe	Xulu	2011/01/26	0822068315
9	Neliswa	Mhlongo	2010/01/15	0725193124
10	Koketso	Zungu	2011/09/14	0625174635

tblActivities contains the activity details

FIELDS	DATA TYPE	DESCRIPTION
ActivityID	INTEGER	A unique autonumber given to each activity; used as the primary key of the table.
Description	TEXT	Description of the activity.
CostPerPerson	DOUBLE	The cost per person for the activity.
ActivityDays	TEXT	The days on which the activity happens.
AllYear	BOOLEAN	A boolean field indicating if the activity happens outside of the holiday times.

The first 10 records of tblActivities are shown below:

ActivityID	Description	CostPerPerson	ActivityDays	AllYear
1	Bike rides	55	We,Fr	Yes
2	Picnic in the park	75	We,Th,Fr	Yes
3	Stargazing	105	We,Th,Fr	No
4	Kids science experiments	60	Mo,Tu,We,Th,Fr,Sa	No
5	Tennis tournament	50	Mo,Tu,We,Th,Fr,Sa	Yes
6	Make a kaleidoscope	110	Fr,Sa	No
7	Water balloon baseball	75	Mo,We,Fr	Yes
8	Paint rocks	100	We,Fr	No
9	Dig for fossils	65	Mo,We,Fr	No
10	Mountain hiking	80	We,Fr	Yes

tblSignUps contains the details of a signup indicating which children have been signed up to which activities.

FIELDS	DATA TYPE	DESCRIPTION
SignupID	INTEGER	A unique autonumber for each sign-up; used as the primary key of the table.
ActivityID	INTEGER	The activity that was selected. A foreign key to tblActivity.
ChildID	INTEGER	The id of the child that was signed up. A foreign key to tblChildren.
SignUpDate	DATE	The date of the sign-up.

The first 10 records of tblSignUps are shown below:

SignUpID	ActivityID	ChildID	SignUpDate
1	11	20	2023/03/31
2	14	14	2023/04/08
3	4	25	2023/12/27
4	2	16	2023/10/04
5	7	2	2024/01/06
6	11	6	2023/04/07
7	10	2	2023/09/30
8	7	10	2023/12/31
9	5	6	2023/10/02
10	11	8	2023/03/25

1.1 Display the details of all children registered in alphabetical order of surname. List all the fields in the table.

The first 10 records of output are shown below:

ChildID	Firstname	Lastname	DOB	ParentCellphone
19	Keagan	Coetzee	2013/07/31	0629750551
1	Kaylyn	Furman	2014/06/18	0828842261
16	Gabriela	Gamildien	2009/02/17	0627275532
15	Chris	Grobler	2013/06/19	0824789512
6	Oratile	Hlanti	2007/12/13	0624359939
13	Karabo	Khune	2006/11/16	0621456325
20	Thapelo	Lebusa	2006/10/07	0725764630
21	Thulani	Maela	2010/12/21	0624563214
12	Ben	Makhabane	2008/09/03	0823135973
14	Mosa	Martin	2008/07/25	0725063507

(4)

1.2 Display which activities happen all year and cost R75 or less per person. List all the fields in the table.

The correct output is shown below:

ActivityID	Description	CostPerPerson	ActivityDays	AllYear
1	Bike rides	55	We,Fr	Yes
2	Picnic in the park	75	We,Th,Fr	Yes
5	Tennis tournament	50	Mo,Tu,We,Th,Fr,Sa	Yes
7	Water balloon baseball	75	Mo,We,Fr	Yes

(5)

1.3 Display which activities take place on a Saturday. List all the fields in the table.

The correct output is shown below:

ActivityID	Description	CostPerPerson	ActivityDays	AllYear
4	Kids science experiments	60	Mo,Tu,We,Th,Fr,Sa	No
5	Tennis tournament	50	Mo,Tu,We,Th,Fr,Sa	Yes
6	Make a kaleidoscope	110	Fr,Sa	No
13	Adventure book reading	100	Fr,Sa	No
14	Cardboard boat racing	120	Mo,Tu,We,Th,Fr,Sa	No

(4)

1.4 The company would like to send advertisements to parents of children who will be turning 13 after today this year or who will have birthdays in June or December. Display only the surname of the child, the parent's cellphone number and date of birth for children that meet these criteria.

The correct output is shown below:

Lastname	ParentCellphone	DOB
Furman	0828842261	2014/06/18
Hlanti	0624359939	2007/12/13
Zungu	0625174635	2011/09/14
Grobler	0824789512	2013/06/19
Singh	0727053281	2011/10/22
Tau	0624443974	2011/09/07
Maela	0624563214	2010/12/21

(5)

1.5 Display the top 3 most expensive activities. List all the fields in the table.

The correct output is shown below:

ActivityID	Description	CostPerPerson	ActivityDays	AllYear
14	Cardboard boat racing	120	Mo,Tu,We,Th,Fr,Sa	No
6	Make a kaleidoscope	110	Fr,Sa	No
3	Stargazing	105	We,Th,Fr	No

(5)

1.6 Determine the number of sign-ups and total amount made from the sign-ups for each activity. Display the activity description, number of sign-ups as a field called TotalSignups and the total amount as a field called TotalAmount.

The correct output is shown below:

Description	TotalSignUps	TotalAmount
Adventure book reading	3	300
Beach kite flying	5	325
Bike rides	1	55
Cardboard boat racing	5	600
Dig for fossils	3	195
Kids science experiments	2	120
Make a kaleidoscope	2	220
Mountain hiking	5	400
Paint rocks	5	500
Picnic in the park	1	75
Tennis tournament	2	100
Water balloon baseball	4	300

(7)

1.7 A change to the format of how the parents' cellphone numbers are stored must be made. Write a query that will change all cellphone numbers to the following format:

(first three digits)<space>next 3 digits<dash>remaining digits

Example of the old format and the changed new format for the first 5 records:

ParentCellphone	NewFormat
0828842261	(082) 884-2261
0627242377	(062) 724-2377
0725884662	(072) 588-4662
0625435781	(062) 543-5781
0739347124	(073) 934-7124

(6)

1.8 Parents need to be invoiced for the total amount owed for all the activities that their child has been signed up for. Invoices for parents who owe more than 250 will be billed first. Display the last name and first name of the child, and the total amount owed for the activity sign-ups for all those who owe more than 250.

The correct output is shown below:

Lastname	Firstname	AmountOwed
Lebusa	Thapelo	265
Maela	Thulani	255
Mothwa	Lyle	340
Williams	Pride	275
Xulu	Ongeziwe	265

(8)

1.9 Some parents would like to make some early sign-ups for this year for some of the same activities their children did last year. Write a query that will add sign-ups for children with ChildIDs of 1,5,7 and 9 for the same activities that they were signed up for but with a sign-up date of 2024/12/10.

The following 6 records were added after this query was run:

SignUpID	ActivityID	ChildID	SignUpDate
39	8	7	2024/12/10
40	9	1	2024/12/10
41	10	9	2024/12/10
42	14	7	2024/12/10
43	14	7	2024/12/10
44	6	5	2024/12/10

(6)

50 marks

SECTION B OBJECT ORIENTATED PROGRAMMING

SCENARIO:

Your teacher has approached you to help create a system to manage the different school clubs and external clubs that are hosted at the school. Each club has a teacher in charge and a specified meeting time agreed on by the members of the club.

The external clubs have both a teacher in charge and a manager who runs the club. All clubs have a basic cost of R135.75 that the school will add to the student's account. However, external clubs charge a rate based on the basic cost that the school assigns to clubs.

The teacher in charge of all the clubs has provided you with the text file **clubs.txt**, which contains all the information on school clubs and external clubs.

Below are the first 10 lines of the text file:

```
Book Club#14:45#J Strydom
Dance Club#14:30#K Chawane#O Ngubane#160
History Club#15:00#O Nche
Pottery Club#14:30#N Taljaard#R Snyman#105
Creative Writing Club#16:30#L Potgieter
Investment Club#15:15#D Vermeulen#S van der Merwe#115
Improv Club#14:45#M van Staden
Mathletes Club#16:15#K Pretorius
Art History Club#14:15#N Smith
Foreign Languages Club#16:45#M Marx
```

Each line in the text file represents a school club or an external club.

SchoolClub

The details for a school club are stored as follows:

<club name>#<meeting time >#<teacher in charge>

- **Club name:** a string storing the name of the club
- **MeetingTime:** a time set for the club members to meet in the format HH:mm
- **Teacher:** the initial and surname of the teacher in charge of the club

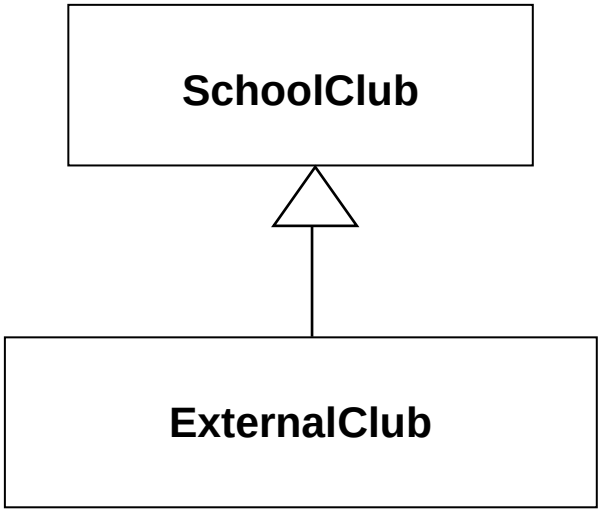
ExternalClub

The details for an external club are stored in the text file as follows:

<club name>#<meeting time >#<teacher in charge>#<manager>#<rate>

- **Club name:** a string storing the name of the club
- **MeetingTime:** a time set for the club members to meet in the format HH:mm
- **Teacher:** the initial and surname of the teacher in charge of the club
- **Manager:** the initial and surname of the non-staff member who offers the club
- **Rate:** the rate that the club charges which is a percentage of the basic club cost

You will need to create two classes: the parent class **SchoolClub** and the child class **ExternalClub**, as shown in the diagram below:



QUESTION 2

Use the class diagram below to create the **SchoolClub** class. This class will be used to create objects to store the details of the school club. The class diagram below indicates the required fields and methods.

SchoolClub
Fields: – clubName : string – meetTime : Time – teacher : string + CLUBCOST : real = 135.75
Methods: + Constructor(inCN : string, inMT : Time, inTR : string) + getClubName() : string + getMeetTime() : Time + getTeacher() : string + toString() : string

2.1 Create a new class called **SchoolClub** with **clubName**, **meetTime** and **teacher** fields as indicated above. Use an appropriate time object to store the **meetTime** field. (4)

2.2 Add one **constant static/class** field, **CLUBCOST,** as shown in the diagram. (2)

2.3 Code a constructor method for the class that will accept the string **inCN** for the **clubName** field, a time named **inMT** representing the **meetTime** field, and a string named **inTR** representing the teacher field. Use these parameters to assign values to the associated fields. (3)

2.4 Create accessor/getter methods for the **clubName**, **meetTime** and **teacher** fields of the class. (3)

2.5 Add a **toString** method to the class that will return a **string** containing the **clubName**, **meetTime**, and **teacher** in the format below.

clubName<space>@<space>**meetTime**<space>Teacher:<space>**teacher**

Example:
BookClub @ 14:45 Teacher: J Strydom

(4)
[16]

QUESTION 3

Use the class diagram below to create a class called **ExternalClub**. This class will inherit from the **SchoolClub** class and will be used to create objects to store external club details. The class diagram below indicates the required fields and methods.

ExternalClub
Fields: – manager : string – rate : real
Methods: + Constructor(inCN : string, inMT : Time, inTR : string, inMR: string, rate : real) + getExternalClubCost() : real + toString() : string

3.1 Create a new class called **ExternalClub** that inherits from **SchoolClub**. (2)

3.2 Add the **manager** and **rate** fields as shown in the diagram. (1)

3.3 Code a constructor method to initialise the **clubName**, **meetingTime** and **teacher** fields of the **SchoolClub** parent class and the fields **manager** and **rate** of the **ExternalClub** child class. (4)

3.4 Code a method called **getExternalClubCost** to calculate and return the external club cost as a real number. The method should calculate the cost of the external club based on the following formula:

External club cost = CLUBCOST × (rate / 100)

Round this result to two decimal places and return this value as a real number. (6)

3.5 Code a **toString** method that will override the parent class's **toString** method. Add the **manager** field and the **cost of the external club** to the parent's **toString** in the format below.

clubName<space>@<space>**meetTime**<space>Teacher:<space>**teacher**<space>Manager: <space>**manager**<space>**Rcost of external club**

Example:
Film Club @ 16:30 Teacher: K Smith Manager: P du Toit R190.05

(4)
[17]

QUESTION 4

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

4.2 Add two instance variables to this class:
- An **array** called **cArr** to store 20 **SchoolClub** or **ExternalClub** objects.
- An integer called **size** to record the number of objects added to the array.

These two variables should not be accessible outside this class. (4)

4.3 Code a constructor method that will read the contents of the file **clubs.txt**. Each line of the text file contains either the details for a school club or external club in the following format:
<club name>#<meeting time >#<teacher in charge> for a school club
Or
<club name>#<meeting time >#<teacher in charge>#<manager>#<rate> for an external club.

The constructor should do the following:
- Read each line from the file and extract the data.
- Instantiate either a **SchoolClub** or **ExternalClub** object based on the line's extracted data.
- Add the **SchoolClub** or **ExternalClub** object to the next available position in the array.
- Increase the **size of the array**. (11)

4.4 Code a **toString** method to combine the values of each object in the array **cArr** into a string using the **toString** methods created in Questions 2.5 and 3.5. The details of each club object should be displayed on a new line. (4)

4.5 Create a method named **sortByMeetTime** to sort the objects in the array **cArr** in chronological order of the meeting time i.e. from the earliest meeting time to the latest meeting time. (8)
[28]

QUESTION 5

5.1 Create a **ClubUI** class with a simple text-based user interface and output. (1)

5.2 Instantiate a **ClubManager** object named **cm** in an appropriate place in this class. (2)

5.3 Display all the details of the clubs using the **ClubManager** object **cm**. (2)

```
Book Club @ 14:45 Teacher:J Strydom
Dance Club @ 14:30 Teacher:K Chawane Manager: O Ngubane R217.2
History Club @ 15:00 Teacher:O Nche
Pottery Club @ 14:30 Teacher:N Taljaard Manager: R Snyman R142.54
Creative Writing Club @ 16:30 Teacher:L Potgieter
Investment Club @ 15:15 Teacher:D Vermeulen Manager: S van der Merwe R156.11
Improv Club @ 14:45 Teacher:M van Staden
Mathletes Club @ 16:15 Teacher:K Pretorius
Art History Club @ 14:15 Teacher:N Smith
Foreign Languages Club @ 16:45 Teacher:M Marx
Film Club @ 16:30 Teacher:K Smith Manager: P du Toit R190.05
Cooking Club @ 15:30 Teacher:M Orie Manager: F Malherbe R203.63
Chess Club @ 15:00 Teacher:V Koch
Video Game Club @ 16:15 Teacher:I Griesel Manager: J Kleyn R142.54
Photography Club @ 16:15 Teacher:E van der Berg Manager: R Nketsa R156.11
Robotics Club @ 14:45 Teacher:J Wiese
Soup Kitchen Club @ 16:00 Teacher:S Kitshoff Manager: B Mbonambi R142.54
```

5.4 **Sort** and redisplay the clubs by their meeting time from the earliest time to the latest as indicated in the sample output below: (2)

[7]

```
Art History Club @ 14:15 Teacher:N Smith
Pottery Club @ 14:30 Teacher:N Taljaard Manager: R Snyman R142.54
Dance Club @ 14:30 Teacher:K Chawane Manager: O Ngubane R217.2
Improv Club @ 14:45 Teacher:M van Staden
Book Club @ 14:45 Teacher:J Strydom
Robotics Club @ 14:45 Teacher:J Wiese
Chess Club @ 15:00 Teacher:V Koch
History Club @ 15:00 Teacher:O Nche
Investment Club @ 15:15 Teacher:D Vermeulen Manager: S van der Merwe R156.11
Cooking Club @ 15:30 Teacher:M Orie Manager: F Malherbe R203.63
Soup Kitchen Club @ 16:00 Teacher:S Kitshoff Manager: B Mbonambi R142.54
Photography Club @ 16:15 Teacher:E van der Berg Manager: R Nketsa R156.11
Mathletes Club @ 16:15 Teacher:K Pretorius
Video Game Club @ 16:15 Teacher:I Griesel Manager: J Kleyn R142.54
Creative Writing Club @ 16:30 Teacher:L Potgieter
Film Club @ 16:30 Teacher:K Smith Manager: P du Toit R190.05
Foreign Languages Club @ 16:45 Teacher:M Marx
```

QUESTION 6

Your teacher needs you to find out how many of the clubs' meeting times have clashes with one another so that they can determine the smallest number of venues to use.

Take note of the following:

- School clubs will be placed in separate venues from external clubs. Therefore, you only need to check school clubs for clashes against other school clubs and external clubs for clashes against other external clubs.
- Each club meeting runs for 45 minutes.

For example
Art History is a school club and has a start meeting time of 14:15. The end time will therefore be 15:00 (45 minutes later).

Art History will therefore clash with:

- the Improv Club and Book Club which both start at 14:45 and
- the Chess club and History Club which both start at 15:00

Since Art History is a school club, there will be no clash with Pottery or Dance because Pottery and Dance are external clubs which would be in another venue.

6.1 In the **ClubManager** class, code a method called **checkClubs**. This method must return a list showing **how many** clashes each club has with other clubs. See sample output in 6.2.

The method must do the following:

- Sort the clubs by their meeting time.
- Create a list to store the clashes for school clubs and
  create a separate list to store the clashes for external clubs as follows:
  - Check if it is a school club or an external club.
  - Check against the other clubs of the same type if the meeting times would clash, given that meetings last for 45 minutes.
  - Count the number of clashes that occur for the club.
  - Append the name of the club followed by a new line and the number of clashes to the respective list (either the school club list or the external list).
- If the club is an external club, a venue cost must be calculated and added to the list as a new line below the number of clashes. Use the following formula to calculate the venue cost, rounded off to two decimal places:

  Venue cost = External Club Cost * 15%
  Note: you need not add a separate method to calculate the venue cost.

- If there are no clashes for a specific club, then the line "==NO CLASHES==" should be added below the club name as shown in 6.2.
  The method must return a string showing the school club clashes first followed by the external club clashes. Add an appropriate heading for each list displayed (see sample output in 6.2).

(30)

6.2 In the **ClubUI** class, call the **checkClubs** method to display the club clashes for the different types of clubs.

Your output should appear as follows:

```
School Clubs
Art History Club
-Clashes with 5 other clubs

Improv Club
-Clashes with 4 other clubs

Book Club
-Clashes with 3 other clubs

Robotics Club
-Clashes with 2 other clubs

Chess Club
-Clashes with 1 other clubs

History Club
==NO CLASHES==

Mathletes Club
-Clashes with 2 other clubs

Creative Writing Club
-Clashes with 1 other clubs

Foreign Languages Club
==NO CLASHES==


External Clubs
Pottery Club
-Clashes with 2 other clubs
Venue Cost: R21.38

Dance Club
-Clashes with 1 other clubs
Venue Cost: R32.58

Investment Club
-Clashes with 2 other clubs
Venue Cost: R23.42

Cooking Club
-Clashes with 3 other clubs
Venue Cost: R30.54

Soup Kitchen Club
-Clashes with 3 other clubs
Venue Cost: R21.38
```

```
Photography Club
-Clashes with 2 other clubs
Venue Cost: R23.42

Video Game Club
-Clashes with 1 other clubs
Venue Cost: R21.38

Film Club
==NO CLASHES==
Venue Cost: R28.51
```

(2)
[32]

100 marks

Total: 150 marks