NATIONAL SENIOR CERTIFICATE EXAMINATION
MAY 2022

INFORMATION TECHNOLOGY: PAPER I

Time: 3 hours 150 marks

PLEASE READ THE FOLLOWING INSTRUCTIONS CAREFULLY

1. This question paper consists of 18 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 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, 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 might find an alternative method of representing the data that is 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/output 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 for the printing of your work.

17. You should be 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:
    ScientiaUniversity.mdb
    ScientiaUniversity_JavaDB.sql
    ScientiaUniversity_MySQL.sql
    SQLAnswerSheet.rtf
    SQLBrowser.exe

    Section B:
    candidates.txt
    researchPapers.txt

SCENARIO:

Scientia University is a new online university aimed at providing South African students with a world-class, affordable and accessible tertiary education. The university has four main departments named MatSci (Mathematics and Science), BusCom (Business and Computer Sciences), HumArt (Humanities and Arts), LawMed (Law and Medicine). The aim is to produce academic researchers who work on innovative research projects producing academic research papers.

You are a recent addition to the Scientia's database administration team. The database called **ScientiaUniversity** contains the details of the many researchers, each assigned to a research project. Each research project has multiple researchers and a lead researcher. In addition, each research project will have an allocated supervisor to oversee and guide the research project. A supervisor can oversee many research projects but is limited to a maximum number of projects to ensure the supervisor is not overworked.

SECTION A SQL

QUESTION 1

The data on the researchers, supervisors, research projects and allocations are stored in a database called **ScientiaUniversity**.

tblResearchers contains all the information about the researchers.

FIELDS	DATA TYPE	DESCRIPTION
ResearcherID	INTEGER	A unique auto-numbered identification number for a researcher.
Firstname	TEXT	The first name of the researcher.
Surname	TEXT	The surname of the researcher.
Department	TEXT	The department to which the researcher belongs.

The first 10 records of tblResearchers:

ResearcherID	Firstname	Surname	Department
1	Gloria	Mofokeng	MatSci
2	Tshegofatso	Theron	LawMed
3	Jean	Kumalo	HumArt
4	Jennifer	Stander	BusCom
5	Cheryl	Snyman	BusCom
6	Clint	Mazibuko	BusCom
7	Arthur	Sibeko	BusCom
8	Grace	Ross	HumArt
9	Nonhlanhla	Mazibuko	BusCom
10	Eric	Mathebula	MatSci

tblSupervisors contains all the information about the supervisors.

FIELDS	DATA TYPE	DESCRIPTION
SupervisorID	INTEGER	A unique auto-numbered identification number for a supervisor.
SupervisorName	TEXT	The name and surname of the supervisor.
Department	TEXT	The department to which the supervisor belongs.
MaxCapacity	INTEGER	The maximum number of research projects the supervisor can oversee.

All the records of tblSupervisors:

SupervisorID	SupervisorName	Department	MaxCapacity
1	Charmaine Kumalo	BusCom	2
2	Janine Liebenberg	MatSci	3
3	Ayanda Thompson	BusCom	4
4	Thato de Kock	MatSci	3
5	Fikile Naicker	LawMed	2
6	Stephen van Dyk	BusCom	3
7	Xolani Beukes	HumArt	1
8	Siphiwe Mthembu	BusCom	1
9	Sam de Jager	HumArt	4
10	Mpho Geldenhuys	BusCom	4

tblResearchProjects contains all the information about the research projects.

FIELDS	DATA TYPE	DESCRIPTION
ResearchProjectID	INTEGER	A unique auto-numbered identification number for a research project.
ProjectName	TEXT	The name of the research project.
SupervisorID	INTEGER	A foreign key to the tblSupervisors that indicates which supervisor has been assigned to oversee this research project.
DateStarted	DATE	The date the project was started.
GrantAmount	INTEGER	The total amount of money granted by the university towards the project.

The first 10 records of tblResearchProjects

ResearchProjectID	ProjectName	SupervisorID	DateStarted	GrantAmount
1	Violet Liberty	4	2016/11/30	150000
2	Steel Packer	5	2018/01/03	240000
3	Feisty Cobra	3	2016/10/23	210000
4	Dapper Crown	4	2018/07/15	80000
5	Late Hawk	3	2017/07/31	50000
6	Gifted Neptune	5	2019/11/19	160000
7	Dark Luna	4	2016/03/05	110000
8	Twilight Lotus	8	2017/03/03	60000
9	Arctic Pawn	9	2018/08/11	80000
10	Exotic Flower	9	2018/10/10	160000

tblAllocations provides a link between the researcher and their research project.

FIELDS	DATA TYPE	DESCRIPTION
ResearcherID	INTEGER	Part of the primary key for this table and a foreign key to the tblResearchers.
ResearchProjectID	INTEGER	Part of the primary key for this table and a foreign key to the tblResearchProject.
DateAssigned	DATE	The date that the research was assigned.
isLead	BOOLEAN	Indicates if the allocated researcher is the lead researcher for the research project.

The first 10 records of the table

tblAllocations			
ResearcherID	ResearchProjectID	DateAssigned	IsLead
1	2	2018/11/06	No
1	13	2019/03/11	No
1	15	2018/09/11	No
2	11	2018/11/20	Yes
2	14	2016/09/26	Yes
2	19	2017/08/24	Yes
2	21	2017/04/30	No
3	3	2016/10/23	Yes
3	12	2017/06/04	No
3	28	2017/07/30	No

1.1 Display all the information about the researchers sorted first by department then by surname. *Only the first five records are shown below.*

ResearcherID	Firstname	Surname	Department
18	Daniel	Khan	BusCom
12	Antoinette	Kumalo	BusCom
9	Nonhlanhla	Mazibuko	BusCom
6	Clint	Mazibuko	BusCom
7	Arthur	Sibeko	BusCom

(4)

1.2 Display a list showing all the information about supervisors who are in either the MatSci or BusCom departments and have a **MaxCapacity** of 3 or more. *The correct output is shown below.*

SupervisorID	SupervisorName	Department	MaxCapacity
2	Janine Liebenberg	MatSci	3
3	Ayanda Thompson	BusCom	4
4	Thato de Kock	MatSci	3
6	Stephen van Dyk	BusCom	3
10	Mpho Geldenhuys	BusCom	4

(5)

1.3 Show the average grant amount given to research projects that were started between the years 2016 and 2018 inclusive. Name this field **AvgGrant**. Round the result to two decimal places. *The correct output is shown below.*

AvgGrant
147826.09

(6)

1.4 Show a count of the number of research projects each supervisor is overseeing. Show the **SupervisorID** and the number of assigned projects as a field named **TotalResearchProjects**. *The correct output is shown below.*

SupervisorID	TotalResearchProjects
1	3
3	4
4	4
5	3
6	5
7	1
8	3
9	4
10	3

(5)

1.5 The departments would like to know the total grant money allocated to projects per year. Write a query to show the year of the project's start date and the total amount of grants issued that year as a field called **GrantsGiven.** List the years that have a total of 500000 (five hundred thousand) or more issued in grants. *The correct output is shown below.*

Years	GrantsGiven
2015	570000
2016	1610000
2017	800000
2018	990000

(8)

1.6 A list of all the lead researchers must be given to each supervisor. Write a query that will show the **ProjectName**, the **FullName** of the researcher, the **SupervisorName** and the **DateStarted** fields. Sort the list by the name of the supervisors in alphabetical order. *The first 5 rows are shown below.*

ProjectName	FullName	SupervisorName	DateStarted
Late Hawk	Nonhlanhla Mazibuko	Ayanda Thompson	2017-07-31
Feisty Cobra	Jean Kumalo	Ayanda Thompson	2016-10-23
Swift Beehive	Cheryl Snyman	Ayanda Thompson	2018-10-24
Calm Supernova	Nonhlanhla Mazibuko	Ayanda Thompson	2016-10-30
Misty Wing	Mike Wilson	Charmaine Kumalo	2017-05-29

(8)

1.7 Some of the researchers have been approached to become supervisors. Write a query that will add the researchers with **ResearcherID**s between 17 and 20 inclusive as new supervisors. They must be given a **MaxCapacity** of 4 and should use their **firstname** and **surname** separated by a space as their **SupervisorName**. *The records added to the table after the query has been run are shown below.*

SupervisorID	SupervisorName	Department	MaxCapacity
11	Shaun Tshabalala	LawMed	4
12	Daniel Khan	BusCom	4
13	Juan Jooste	MatSci	4
14	Simone Green	LawMed	4

(8)

1.8 Some students have complained that the department names are a bit confusing. The university would like each department name to include the "and" symbol ("&"). Write a query that will change each department name to now include the " & " symbol in the name with spaces on either side of the "&" symbol. *An example of what the table will look like after the query has been run is given below.*

SupervisorID	SupervisorName	Department	MaxCapacity
1	Charmaine Kumalo	Bus & Com	2
2	Janine Liebenberg	Mat & Sci	3
3	Ayanda Thompson	Bus & Com	4
4	Thato de Kock	Mat & Sci	3
5	Fikile Naicker	Law & Med	2
6	Stephen van Dyk	Bus & Com	3
7	Xolani Beukes	Hum & Art	1
8	Siphiwe Mthembu	Bus & Com	1
9	Sam de Jager	Hum & Art	4
10	Mpho Geldenhuys	Bus & Com	4
11	Shaun Tshabalala	Law & Med	4
12	Daniel Khan	Bus & Com	4
13	Juan Jooste	Mat & Sci	4
14	Simone Green	Law & Med	4

(6)

50 marks

SECTION B OBJECT-ORIENTATED PROGRAMMING

The university would like your help with creating a system to monitor the research papers that doctoral candidates (referred to as candidates) submit to earn their doctorates. Each research paper will be submitted to the head of the department for that particular research paper. The department heads will review the research paper and if it meets the minimum requirements then the candidate will be awarded their doctoral degree. There are a limited number of slots for candidates each year and the university only accepts 20 candidates in total for all departments.

In addition, any peer-reviewed (the paper has been evaluated by another candidate) research paper that includes field work, exceeds the dean's minimum number of pages of 200 and meets the department's requirements will have their research paper added to the Dean's List for that year.

The program you need to create requires the following base classes:

Candidate Class

A candidate is a university student attempting to earn his or her doctoral degree by completing and submitting a research paper.

- **studentID** – the unique student number for the candidate.
- **name** – the full name of the candidate.
- **year** – the academic year for the candidate.

The details of candidates are stored in a text file called **candidates.txt** with each line representing the data of an individual candidate in the following format:

<studentID> # <name> # <year>

The first 10 lines of the text file are shown below.

```
2014969#Ayanda Robertson#8
2016618#Pieter Evans#6
2016439#Gail Ntombela#6
2015846#Lloyd De Klerk#7
2018111#Lindiwe Kleynhans#4
2016564#Albert Cronje#6
2017619#Fikile Bezuidenhout#5
2016281#Garth Marx#6
2017492#Mark Schoeman#5
2014181#Constance Swartz#8
```

ResearchPaper Class

This class represents the research paper that a candidate must submit to receive their doctoral degree. The departments use a shorthand requirements code (**reqCode**) that combines the number of pages in the research papers with a F or T to indicate if the paper has been peer reviewed and another F or T to indicate if the paper includes field research.

- **title** – the title of the research paper.
- **department** – the department that the research paper must be submitted to.
- **meetsMinRequirements** – if this paper meets the department's minimum requirements.
- **reqCode** – the shorthand code containing all additional details of the research paper.
- **studentID** – the studentID of the candidate who is the author of the research paper.

A text file called **researchPapers.txt** contains the details of the research papers submitted by each candidate.

The first five lines of the text file are shown below.

```
A Discussion on Naturalistic Theory#HumArt#FALSE#81FF#2014811
Agricultural Law in South Africa#LawMed#TRUE#150TT#2018402
Understanding Random Sampling#MatSci#FALSE#126FT#2018111
Homelessness in South Africa#HumArt#FALSE#92TT#2017436
Romanticism in the Enlightenment Era#HumArt#FALSE#102FF#2016871
```

*From the file you can see that the values of the **reqCode** field for the first three research papers are:*

81FF = 81 pages, not peer reviewed, no field research

150TT= 150 pages, peer reviewed, field research included

126FT = 126 pages, not peer reviewed, field research included

QUESTION 2

Use the class diagram below to create a new class named **Candidate**. This class will be used to store the details of a candidate.

Candidate
Fields - studentID : string - name : string - year : integer
Methods + Constructor (inSID : string , inNE : string , inYR : integer) + getStudentID() : string + getName() : string + getYear() : integer + toString() : string

2.1 Create a new class named **Candidate** with the **studentID**, **name** and **year** fields as shown in the class diagram. (3)

2.2 Code a parameterised constructor method for the class that will assign the values to the fields of the class. Use the parameter list provided in the class diagram. (4)

2.3 Add an accessor method for each of the fields of the class. (2)

2.4 Code a **toString** method that will return a string containing all the fields of the class in the following format:

```
name<space>"["studentID"]"<space>"year"<space>year
```

For example:

```
Ayanda Robertson [2014969] year 8
```

(4)

[13]

QUESTION 3

Use the class diagram below to create a new class named **ResearchPaper**. This class will be used to store the details of a research paper.

ResearchPaper
Fields - title : string - department : string - meetsDeptRequirements : boolean - reqCode : string - studentID : string + DEANMINPAGES : integer = 200
Methods + Constructor (inTE : string, inDT : string, boolean : inMDR inRC : string, inSID : string) + getTitle() : string + getDepartment() : string + getStudentID() : string + getMeetsDeptRequirements() : boolean + onDeanList() : boolean + toString() : string

3.1 Create a new class named **ResearchPaper** with the **title**, **department**, **meetDeptRequirements, reqCode** and **studentID** fields as shown in the **class** diagram. (3)

3.2 Create the constant **DEANMINPAGES** as shown in the class diagram. The constant can be static or non-static. (3)

3.3 Code a parameterised constructor method for the class that will accept parameters for the **title**, **department**, **meetDeptRequirements, reqCode** and **studentID** fields. Use the parameter list provided in the class diagram. (4)

3.4 Create accessor methods for the **title**, **department**, **meetDeptRequirements** and **studentID** fields. (2)

3.5 Create a **toString** method that will return a string containing the values of the fields in the **ResearchPaper** class. It should also include if the research paper meets the department requirements. The string should be formatted as follows:

```
"Title:"<space>title<newline>
"Author:"<space>studentID<newline>
"Meets"<space>department"'s"<space>"requirements:"<space>
<meetsDeptRequirements>
```

For example:
Example 1:

```
Title: A Discussion on Naturalistic Theory
Author: 2014811
Meets HumArt's requirements: false
```

Example 2:

```
Title: Agricultural Law in South Africa
Author: 2018402
Meets LawMed's requirements: true
```

(5)
[17]

You will note that the method **onDeanList()** has not been created yet. This will be dealt with in Question 6 later in the paper.

QUESTION 4

4.1 Create a class named **CandidatePaperManager.** (1)

4.2 Create an array called **cArr** to store exactly 20 **Candidate** objects. This array should not be accessible outside the class. (2)

4.3 Code a constructor method that will read the contents of the text file named **candidates.txt** that contains the information for each candidate. There are exactly 20 lines in the file representing the 20 candidates approved by the departments. Your method should do the following:

- Check if the file **candidates.txt** exists. Display an error message if the file does not exist.
- Open the file for reading.
- Loop through all the lines of the text file. In each iteration of the loop:
  - Read the line and split the data for a **Candidate** into the separate parts.
  - Create a **Candidate** object using the split data and add the object to the array called **cArr** in the next available position. (9)

4.4 Code a method named **displayAllCandidates** that will return a string containing the information about all the candidate objects stored in the array called **cArr**. The method must make use of the **toString** method you coded in Question 2.4. Each candidate's information should appear on a new line. (5)

4.5 Write code to create a method named **sortByYear.** This method should sort the candidates in the array according to their academic year in descending order. (8)

[25]

QUESTION 5

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

5.2 Declare and instantiate a **CandidatePaperManager** object in an appropriate place in the code. (1)

5.3 Write code that will call the appropriate method in the **CandidatePaperManager** class to **display** a list of all the **Candidate** objects. *Sample output of the first 5 and last 4 candidates are shown below:*

```
Ayanda Robertson [2014969] year 8
Pieter Evans [2016618] year 6
Gail Ntombela [2016439] year 6
Lloyd De Klerk [2015846] year 7
Lindiwe Kleynhans [2018111] year 4
...
Ntokozo Francis [2015881] year 7
Clint Mazibuko [2016334] year 6
Arthur Sibeko [2015413] year 7
Grace Ross [2014950] year 8
```

(1)

5.4 **Add code to sort** all the candidates into descending order by their year and **display** the sorted list. *Sample output of the first 5 and last 4 candidates are shown below. Note the order may differ slightly depending on the type of sort method used:*

```
Ayanda Robertson [2014969] year 8
Constance Swartz [2014181] year 8
Terence Joubert [2014811] year 8
Grace Ross [2014950] year 8
...
Mark Schoeman [2017492] year 5
Fikile Bezuidenhout [2017619] year 5
Natalie Snyman [2018402] year 4
Lindiwe Kleynhans [2018111] year 4
```

(2)
[5]

QUESTION 6

Return to the **ResearchPaper** class to create the method to determine if a research paper is eligible for the Dean's List.

Add a method named **onDeansList** to the **ResearchPaper** class that will return a boolean value of **true** if the research paper meets the department's minimum requirements, is peer reviewed, includes field research and has more pages than the constant value stored in the field DEANMINPAGES. You must make use of the constant value to earn the full mark allocation.

[7]

QUESTION 7

The departments must review the research papers submitted by the candidates.

7.1 In the **CandidateManager** class, add two more instance variables:

- An array called **rpArr** that will be used to store up to 20 **ResearchPaper** objects.
- A counter called **rpSize** to keep track of the number of **ResearchPaper** objects added to the **rpArr**. (2)

7.2 In the **CandidateManager** class, create a method named **reviewResearchPapers**. This method should do the following:

- Read all the data in the text file **researchPapers.txt** and create a **ResearchProject** object called **rp** from each line.
- Add the **ResearchProject** object called **rp** to the array **rpArr** if the object meets the department's minimum requirements.
- Return a string containing two sections: one for those papers that meet the minimum requirements followed by those that do not. Use the headings "**Accepted**" and "**Rejected**". (11)

7.3 Call the **reviewResearchPapers** method in **CandidatePaperUI** class and display the list of accepted and rejected research papers. *The correct output is shown below:*

```
Accepted
Title: Agricultural Law in South Africa
Author: 2018402
Meets LawMed's requirements: true
Title: Nature of Business Competition
Author: 2014950
Meets BusCom's requirements: true
Title: Behaviour of Currencies
Author: 2014969
Meets BusCom's requirements: true
Title: Copyright Infringements in Music
Author: 2017619
Meets LawMed's requirements: true
Title: Fate in Greek Mythology
Author: 2016564
Meets HumArt's requirements: true
Title: Media versus Materialism
Author: 2016439
Meets HumArt's requirements: true
Title: Understanding Business
Author: 2017433
Meets BusCom's requirements: true
Title: Mathematics in Artificial Intelligence
Author: 2016281
Meets MatSci's requirements: true
```

```
Rejected
Title: A Discussion on Naturalistic Theory
Author: 2014811
Meets HumArt's requirements: false
Title: Understanding Random Sampling
Author: 2018111
Meets MatSci's requirements: false
Title: Homelessness in South Africa
Author: 2017436
Meets HumArt's requirements: false
Title: Romanticism in the Enlightenment  Era
Author: 2016871
Meets HumArt's requirements: false
Title: A Study of Light Painting
Author: 2014181
Meets HumArt's requirements: false
Title: Tax Reforms Effects on the Economy
Author: 2017810
Meets LawMed's requirements: false
Title: Review of the Great Depression
Author: 2015881
Meets BusCom's requirements: false
Title: Ancient Greek Theater
Author: 2015846
Meets HumArt's requirements: false
Title: Understanding the Placi Effect
Author: 2017492
Meets MatSci's requirements: false
Title: Muscle systems in the Human Body
Author: 2016618
Meets MatSci's requirements: false
Title: Principals of Multiculturalism
Author: 2016334
Meets HumArt's requirements: false
Title: The Theft Act
Author: 2015413
Meets LawMed's requirements: false
```

(1)
[14]

QUESTION 8

Once the review process is complete, the university will host a ceremony where the successful candidates will receive their doctoral degrees. For each department show the name of the candidate, the title of their research paper and if they made the Dean's List for that year.

8.1 Add a method to the **CandidateManager** class named **doctoralCandidates**. This method should return a string containing the name of the successful candidates in that department and the title of their research paper on the next line. If the candidate's research paper meets the requirements for the Dean's List, then a "**(D)**" should appear next to the candidate's name. Add a blank line after each candidate in the list.

To organise the list into departments use a string called **depts** with the value "MatSci#BusCom#HumArt#LawMed" which includes all the departments separated with a "#".

The university will need hard copies of the list to print for the day of the ceremony and for record keeping. Save the complete list to a text file named **doctors.txt**.

You have been asked to include the current date and time in the format YYYY/MM/dd HH:mm:ss at the top of this text file. (17)

8.2 Call the **doctoralCandidates** method in **CandidatePaperUI** class and display the list of doctoral candidates and their research papers. T*he correct output is shown below:*

```
2021/10/18 12:06:53
MatSci
Garth Marx (D)
Mathematics in Artificial Intelligence

BusCom
Grace Ross
Nature of Business Competition

Ayanda Robertson
Behaviour of Currencies

Tania Chetty
Understanding Business

HumArt
Albert Cronje (D)
Fate in Greek Mythology

Gail Ntombela
Media versus Materialism

LawMed
Natalie Snyman
Agricultural Law in South Africa

Fikile Bezuidenhout (D)
Copyright Infringements in Music
```

(2)
[19]

100 marks

Total: 150 marks