SCHOOL EMBLEM

PRELIMINARY EXAMINATION
SEPTEMBER 2021

INFORMATION TECHNOLOGY: PAPER I

Ti 150 marks

Student Name	

EXAMINERS: E van Aarde (Section A), R Viljoen (Section B)
MODERATORS: S Barber, M Ounaceur, M Walker

PLEASE READ THE FOLLOWING INSTRUCTIONS CAREFULLY

1. This question paper consists of 15 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 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 the 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:
    DigitalNimbus.mdb
    DigitalNimbus_JavaDB.sql
    DigitalNimbus_MySQL.sql SQLAnswerSheet.rtf
    SQLBrowser.exe

    Section B:
    servers.txt
    tage11resultstechnicians.txt

SCENARIO:

SECTION A SQL

QUESTION 1

50 marks

SECTION B

SCENARIO

The Tour de France is an annual men's multiple-stage bicycle race primarily held in France, while also occasionally passing through nearby countries. Like the other Grand Tours (the Giro d'Italia and the Vuelta a España), it consists of 21 stages, each a day long, over the course of 23 days, and it covers an astonishing 3 343 km. It is the oldest of the Grand Tours and generally considered the most prestigious.

Three riders dominated this year's Tour: the defending champion Tadej Pogacar of Slovenia, Jonas Vingegaard of Denmark and Wout van Aert of Belgium. South Africa's Louis Meintjies also finished in the Top 10.

There are four jersey colours awarded after each day's race.

- The current overall Race Leader receives the Yellow Jersey.
- The best Sprinter receives the Green Jersey.
- The King of the Mountain (best rider over the immense climbs) receives the Polkadot Jersey. Like the Green Jersey, riders are awarded points for completing certain sections of the Stage the fastest.
- The Best Young Rider (under 23 years of age) receives the White Jersey.

You are tasked with creating a program that will be able to read a text file containing a Stage's results, and update the current overall standings for the Yellow and Green Jerseys.

The text file "overall.txt" contains data on the current rider standings after Stage 10:

```
T. POGACAR#UAE TEAM EMIRATES#37:11,28#139
L. KÄMNA#BORA - HANSGROHE#37:11,39
J. VINGEGAARD#JUMBO - VISMA#37:12,07
G. THOMAS#INEOS GRENADIERS#37:12,45
A. YATES#INEOS GRENADIERS#37:12,53
```

Each line contains a rider's name, the team he rides for, and his current overall race time. Some riders have an extra field that holds his current points for sprinting. Up until Stage 10, Tadej Pogacar rode 37 hours, 11 minutes and 28 seconds, and have accumulated 139 Sprint points.

The text file "Stage11Results.txt" contains the results for Stage 11:

```
J. VINGEGAARD#JUMBO - VISMA#16:53,02
N. QUINTANA#TEAM ARKEA - SAMSIC#16:54,01
```

```
R. BARDET#TEAM DSM#16:54,12
G. THOMAS#INEOS GRENADIERS#16:54,40
D. GAUDU#GROUPAMA - FDJ#16:55,06
A. YATES#INEOS GRENADIERS#16:55,12
T. POGACAR#UAE TEAM EMIRATES#16:55,53
…
W. VAN AERT#JUMBO - VISMA#17:10,29#20
```

Each line contains a rider's name, the team he rides for, and the time he finished on the day (local time). Some riders have an extra field that shows how many Sprint points he scored on the Stage. Wout van Aert came in at 17:10,29 (ten minutes past five in the afternoon), and scored 20 points for Stage 11's Sprint section.

QUESTION 2

The **Rider** class was designed with the following class diagram. It indicates the properties and methods required. Note the method names and parameter names in the diagram. No additional *public* method or property may be created. You may create additional *private* properties and methods if you need to.

Rider
- name : string - team : string - overall: string - points: integer
+ Constructor(n : string, t : string, o : string) + getName() : string + getTeam() : string + getOverall () : string + setPoints(p: int) + setOverall(start: LocalTime, end: LocalTime) + fileLine() : string + toString() : string

2.1 Create a new class called **Rider**. (1)

2.2 Create the properties as indicated in the class diagram. (3)

2.3 Create a constructor method that will accept parameters to initialise the **name**, **team** and **overall** time properties of the class. The **points** property must be set to 0. (3)

2.4 Create accessor methods for the all the class properties. (2)

2.5 Create a mutator method for the **points** property. It should receive a parameter value *to be added* to the current points value. (2)

2.6 Create the **toString** method to return display data in the format shown below.

<name><tab><team><tab><overall time>

For example:

```
T. POGACAR              UAE TEAM EMIRATES      37:11,28
```

(2)

2.7 Create the **fileLine** method. It should return data that will be written to a file in the format shown below:

<name>#<team>#<overall time>#<points>

```
T. POGACAR#UAE TEAM EMIRATES#37:11,28#139
```

(2)

[15]

QUESTION 3

The Jersey class was designed with the following class diagram. This class is a subclass of the **Rider** class and will be used to create objects that will store Jersey Wearers. The class diagram indicates the properties and methods required. Note the method names and parameter names in the diagram. No additional *public* method or property may be created. You may create additional *private* properties and methods if you need to.

Jersey
- jerseyColour: String + YELLOW_JERSEY : String = "YELLOW JERSEY" + GREEN_JERSEY : String = "GREEN JERSEY"
+ Constructor(n : string, t : string, o : string) + getJersey() : string + setJersey(jC : string)

3.1 Create a new class called **Jersey**. It is a derived class of **Rider**. (2)

3.2 Create the class property as indicated in the class diagram. (2)

3.3 Create the two constant identifiers as indicated in the class diagram. These constants must be visible from outside the class and can be static or non-static. (2)

3.4 Create the constructor to initialise the parent class properties. The class property must be set to a default value. (3)

3.5 Create the indicated accessor and mutator for the class property. (2)

[11]

QUESTION 4

4.1 Create a new class called Controller. (1)

4.2 Create an array called **arr** that can store 200 **Rider** objects and an integer counter to keep track of how many titles are stored in the array. (3)

4.3 Create a new private helper method called **readFile**. The method should read the contents of the text file called "overall.txt" and populate the Rider array you created in the previous question.

- Open the file for reading.
- Loop through the text file and extract the data from each line. Each line contains data for one **Rider** object. Create the object and place it in the array.

(11)

4.4 Create a default constructor for the Controller class and call the **readFile** method to populate the array. (1)

4.5 Create a new method called **listAll** to return all the output of all the objects in the array as a single string, with each object's output on a new line. Add the rider's position as the first data item. (4)

[21]

QUESTION 5

5.1 Create a new class called **IeTourUI** that will provide a simple text-based interface. (1)

5.2 Create a new Controller object. (1)

5.3 Call the appropriate method to display all the riders. (1)

```
1       T. POGACAR      UAE TEAM EMIRATES       37:11,28
2       L. KÄMNA        BORA - HANSGROHE        37:11,39
3       J. VINGEGAARD   JUMBO - VISMA   37:12,07
4       G. THOMAS       INEOS GRENADIERS        37:12,45
5       A. YATES        INEOS GRENADIERS        37:12,53
```

[3]I

QUESTION 6

Return to the **Rider** class.

6.1 Refer to the class diagram in Question 2 and create a new method called **setOverall**. It should receive a rider's start and end time.

Determine the time difference between the two times (that is the time the rider took to complete the Stage) and add it to his overall time: Pogacar's current overall time is 37 hours, 11 minutes and 28 seconds. He took 4 hours, 20 minutes and 53 seconds to complete the new Stage, which leaves him on 41 hours, 32 minutes and 21 seconds.

Update the **overall** field accordingly. (10)

Return to the **Controller** class.

6.2 Create new private helper method called **sort** that will sort the riders based on their overall time, from quickest to slowest. (7)

6.3 Create a new method called **stageResults**. It should receive a file path as a parameter to a text file that will contain a new Stage's results, update each rider's overall time and points, and write the results back to "overall.txt".

- Ensure that the file exists and give an appropriate error message if it does not.
- Open the file and loop through it to extract the data from each line. Match the rider's result for the Stage to the corresponding object in **arr**.
- Call the appropriate method to update the rider's time. The start time for Stage 11 was 12:30.
- Update the rider's points tally if he scored any points during the Stage.
- Sort the array so that the overall results will reflect the riders' new positions after the stage.
- Overwrite the "overall.txt" text file with the current overall standings. (17)

6.4Add the appropriate method calls to **leTourUI** and the **listAll** method to update and redisplay the current standings. (3)

1	J. VINGEGAARD	JUMBO - VISMA	41:30,9
2	R. BARDET	TEAM DSM	41:32,19
3	T. POGACAR	UAE TEAM EMIRATES	41:32,21
4	G. THOMAS	INEOS GRENADIERS	41:32,25
5	N. QUINTANA	TEAM ARKEA - SAMSIC	41:32,42

[37]

QUESTION 7

Add code to the **listAll** method to assign jersey colours to the current leader (rider with the best time) and the rider with the most sprint points.

7.1 Change the **Rider** object with the best time to a **Jersey** object. Use the appropriate class constant from the **Jersey** class. (3)

7.2 Find the rider with the moist points. Change this **Rider** object to a **Jersey** object as well, and use the appropriate class constant. (6)

7.3 Create a list of jersey wearers and add it to the top of the method's output. Look at Screenshot A to see what the required format is. This shows the rider standings after Stage 10. (4)

```
        [ YELLOW JERSEY ]
T. POGACAR      UAE TEAM EMIRATES       37:11,28
        [ GREEN JERSEY ]
W. VAN AERT     JUMBO - VISMA   37:43,07

1       T. POGACAR      UAE TEAM EMIRATES       37:11,28
2       L. KÄMNA        BORA - HANSGROHE        37:11,39
```

(Screenshot A)

```
        [ YELLOW JERSEY ]
J. VINGEGAARD   JUMBO - VISMA   41:30,9
        [ GREEN JERSEY ]
W. VAN AERT     JUMBO - VISMA   42:18,36

1       J. VINGEGAARD   JUMBO - VISMA   41:30,9
2       R. BARDET       TEAM DSM        41:32,19
3       T. POGACAR      UAE TEAM EMIRATES       41:32,21
```

(Screenshot B)

After Jonas Vingegaard's exceptional performance on Stage 11, the Yellow Jersey changed hands. Wout van Aert was still the wearer of the Green jersey, again scoring maximum points for the sprint section. This is shown in Screenshot B.

[13]

TOTAL SECTION B: {100}