PRELIMINARY EXAMINATION 2016

INFORMATION TECHNOLOGY: Paper II

PECANWOOD COLLEGE
Prepared for Life

Grade 12

Examiners: Mr T Bothma, Mr R Viljoen
Moderator: Ms M. Ounaceur, Ms M. Walker

Date: 1 September 2016

Time: 3 hours

120 marks

PLEASE READ THE FOLLOWING INSTRUCTIONS CAREFULLY

1. This question paper consists of 8 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 that will be more efficient considering the questions 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, eg a power failure, when you resume writing your examination, 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.

Section A STRUCTURED QUERY LANGUAGE

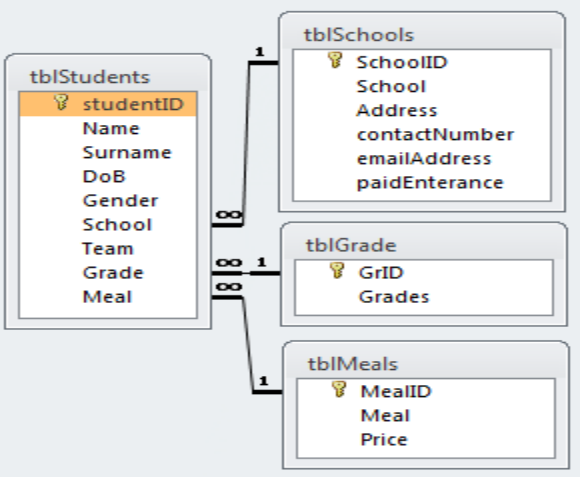
Scenario

All neighbouring schools have been invited to enter the Schools “Brain Power Challenge”. Two teams of each school will compete against each other in attempting to write a program that can creatively outsmart and out survive their opponents.

The theme of the 2016 challenge is centred on the classic 1980's strategy maze-based game, Bomberman.

The organisers of “Brain Power Challenge” wants to keep a record of all the participants.

The database **BPChallenge** contains four tables. The first table **tblStudents** contains the details of each participant including their studentID, Name, Surname, Date of Birth, Gender, the school they represent, team, Grade and Meal preferences number. The second table **tblSchools** contains information on each of schools, including the School ID, name of the school, address, contact telephone number, email and if the entrance fee for the participants are paid. The third table **tblGrade** contains information on the grades (and the grade ID) that will participate in the challenge. The fourth table **tblMeals** contains the Meal ID, meal preferences of the participants and the prices.



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

tblStudents

Field Name	Data Type	Description
studentID	Number	This field assigns a unique ID for each participant.
Name	Short Text	This field contains the paticipant's name.
Surname	Short Text	This field contains the paticipant's surname.
DoB	Date/Time	This field contains the paticipant's date of brirth.
Gender	Short Text	This field contains the paticipant's genger.
School	Number	This field contains the paticipant's school.
Team	Short Text	This field contains the paticipant's team.
Grade	Number	This field contains the paticipant's grade.
Meal	Number	This field contains the paticipant's meal preference.

studentID	Name	Surname	DoB	Gender	School	Team	Grade	Meal
1	James	Simpson	1999/01/17	M	4	A	1	3
2	Peter	Hammer	1999/12/22	M	2	A	4	4
3	Barry	Waldron	2000/06/15	M	1	A	2	4
4	Gary	Hamilton	2000/05/10	M	3	A	2	4
6	Heather	Goldenhersh	2000/05/16	F	2	A	5	2
7	Mackenzie	Crook	1999/09/11	F	1	A	2	1
8	Lynn	Collins	2002/10/20	F	2	A	5	1
9	Mpho	Ndlovu	2002/02/08	M	3	A	4	4

tblSchools

Field Name	Data Type	Description
SchoolID	Number	This field contains a unique ID for each School.
School	Short Text	This field contains the school name.
Address	Short Text	This field contains the school address.
contactNumber	Short Text	This field contains the school telephone number.
emailAddress	Hyperlink	This field contains the school email.
paidEnterance	Short Text	If the entrance fee for the participoants are paid.

SchoolID	School	Address	contactNumber	emailAddress	paidEnterance
1	North Lake Secondary School	125 Robin Avenue	0122359866	admin@northlake.co.za	Yes
2	Penwith High School	85 Roman Drive	0114568695	admin@penwith.za.org	Yes
3	Ridgedale High School	584 Ridge Road	0118562413	principal@ridgedale.com	No
4	Saint Peter's College	11 Drakensberg Road	0118698536	cat@spc.co.za	Yes

tblGrade

Field Name	Data Type	Description (Optional)
GrID	AutoNumber	This field contains a unique ID for each grade. This field is an autonumber field.
Grades	Number	This field contains the grades of the participants.

GrID	Grades
1	8
2	9
3	10
4	11
5	12

tblMeals

Field Name	Data Type
MealID	AutoNumber
Meal	Short Text
Price	Currency

MealID	Meal	Price
1	Halaal	R 35.00
2	Kosher	R 31.00
3	Vegeterian	R 29.00
4	No Preference	R 30.00

QUESTION 1

Write SQL statements for the following:

1.1 Write a query that will list the name, surname and school ID of all participants, sorted by school (ascending) and then by surname (alphabetically). (3)

1.2 Write a query that will list all the details for the female participants. (3)

1.3 Carmel Naido from will be joining her classmate Martin Sebothoma's team.

1.3.1 Use the details of Martin (StudentID 41) to add Carmel's information in a new record. (6)

studentID	Name	Surname	DoB	Gender
40	Carmel	Naidoo	2001/11/26	F

1.3.2 Martin (studentID 41) has decided to spend more time on his schoolwork. Write a query to delete his information from the Students table. (2)

1.4 Write a query to correct the surname of "John Blackburm" to "Blackburn". (2)

1.5 Write a query to display the the number of participants per school per grade. Sort the result desending by school and asending by grade. Rename the calculated field to "Number of Participants ". The output should be as follows: (7)

School	Grades	Number of Participants
Saint Peter's College	8	3
Saint Peter's College	10	4
Saint Peter's College	11	2
Ridgedale High School	8	2
Ridgedale High School	9	1
Ridgedale High School	10	2
Ridgedale High School	11	4

1.6 During the challenge participants are allocated certain tasks like setting up the tables. The oldest students (according to their year of birth) do not have to do any of these tasks. The rest of the students are divided into groups of 4 and allocated a certain task. If there are for example 43 students, ten groups would be formed and 3 groups would have an extra member. Write a query that will calculate the number of groups that will be available to perform tasks during the challenge. (5)

1.7 Each school should have 5 participants in each team. Write a query to display the teams which have less than 5 participants. The result must show how many participants the school must recruit to make a full team. The output should be as follows: (8)

School	Team	To Recruit
Ridgedale High School	A	1
Saint Peter's College	B	1

1.8 The challenge run over 5 days. Calculate the total cost for each meal type for all the participants. (4)

40 marks

Section B OBJECT-ORIENTED PROGRAMMING

The schools are participating in the Spur Schools Mountain Bike series. After the season, riders qualify for the Nationals in October at the ATKV Buffelspoort Resort. Riders qualify based on a rating of their best race during the season.

The following is an excerpt from the text file "Results.txt".

```
Name # Birthdate # School # Time # Race Par Time (min)
Tielman Roos#1999/09/18#Midstream#1h22.34#80
Peter Hammer#1999/12/22#Wonderboom#1h10.10#70
Tielman Roos#1999/09/18#Midstream#1h28.44#80
James Simpson#1999/01/17#Montana#1h30.44#85
```

The text file contains individual riders' results for the three races during the season. Only their time for the race is recorded, as well as a "Par Time" for the race (this is decided by race organisers based on the distance and race difficulty). One rider's details may be recorded up to three times if he participated in all events. Not all riders participate in all the events, but may still qualify for Nationals.

QUESTION 2

2.1 Create a new class called **Race** that will contain two fields for the race time (string) and race par time (integer). (3)

2.2 Create a parameterised **constructor** that will receive a string and integer value for the fields as parameters and assign the values accordingly. (2)

2.3 Create a **toString** method that will return the rider's time in the following format: "Time: 1h59.59" (2)

2.4 Code a new method called **convertTime** that will convert and return the rider's time as a real value. The race times are stored in hours (you may assume this will always be 1 digit), minutes and seconds. This must be converted into minutes and fractions of minutes. e.g. 1h28.45 = 88.75. (4)

2.5 Code a new method called **getTimeDiff** that will return the difference between the achieved time (as a real value) and the par time (**Time – Par time**). (1)

[12]

QUESTION 3

The following class diagram describes the **Rider** class. This class will be used to store the details of a single rider. One property/field is an array of Race objects (the class created in the previous question). The diagram on the next page indicates the properties/fields and methods that are required.

Rider
Properties/Fields: - String name - String dob - String school - *race* : Array of 3 Race objects
Methods: + Constructor(String n, String d, String s) + String getName() + int getAge() + String toString() + void setRace(Race competition) + boolean equals(String n, String d) + double bestResult()

3.1 Use this class diagram to create the new class called **Rider**. (4)

3.2 The **constructor** must assign the values received as parameters to the class fields. The *race* array must be initialised with null objects. (4)

3.3 The **getName** method must return the name and surname of the rider. (1)

3.4 The **getAge** method must return the age the rider will be this year, to determine his age group he will be riding in. (2)

3.5 The **toString** method must return a single string where the fields for a rider are printed on a new line. Remember to use the toString of the Race class.
If Tielman Roos has participated in 2 races, his output will look as follows:

```
Name: Tielman Roos   Age: 17
Race 1 Time: 1h22.34
Race 2 Time: 1h28.44
```

(4)

3.6 Create a new method called **setRace** that will receive a Race object as a parameter. This object must be stored in the *race* array at the first available space (first null object). (4)

3.7 Create a new method called **equals** that receive the name and birthdate of a rider and compares it with the *name* and *dob* fields stored in the object. It should return a single boolean value to indicate if the name and birthdate it received are the same as the stored fields. (3)

3.8 Create a new method called **bestResult** that will look for the rider's best race result, based on his time difference for the race, and return the value. (3)

[25]

QUESTION 4

4.1 Create a new class called **Controller** to store the details of the riders. (1)

4.2 The class must contain two private fields: An array called **riderArr** to store the details of 50 riders, and an integer variable called **count** to store the number of riders in the array. (2)

4.3 Code a method called **findRider.** The method will receive the name (string) and the birthdate (string) of a rider. It must then find the rider in the riderArr. Remember to use the Equals method in the Rider class. Return the Rider object if found. If the rider is not found, the method must return a null value. (5)

4.4 Code a default **constructor** that will assign values to the array using the data in the text file:

4.4.1 For each line in the file you must extract the values and create a Race object. (6)

4.4.2 If the rider object already exists in the array, add the Race object to the correct rider.

If the rider object does not exist in the array, create a new object for a rider and add the Race object. (6)

4.5 Code a **toString** method to return all the riders stored in the array as a single string. Each rider's details must be separated by a blank line. (3)

4.6 Create a list of **riders that qualify** for nationals. This is done by creating a rating for a rider's race by using the following formula:

Time Difference + (rider age * 5)

Only the rider's best race needs to be considered. If his rating is below 80 he will qualify. Remember to use the correct methods created in previous questions. (4)

[27]

QUESTION 5

5.1 Create a new application with a simple interface called **UserInterface**. (1)

5.2 Declare and instantiate a new **Controller** object. (1)

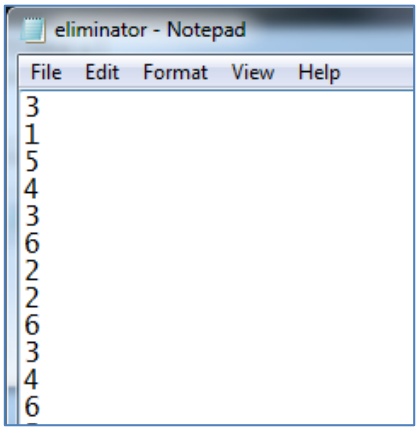
5.3 Call methods to:

Display the riders
Display the list of riders that qualify for Nationals with an appropriate heading. (2)

[4]

QUESTION 6

At last year's Nationals a type of eliminator race was held. The top three riders from the top six schools did a six lap race. As the riders came in after a round, only their school's number was recorded:



(Excerpt from "eliminator.txt")

The schools' numbers are:

1 : Menlopark
2 : Waterkloof
3 : Midstream
4 : Cornwall Hill
5 : Montana
6 : Wonderboom

The winning school is determined by the number of rounds the school has done. As soon as the first school has completed 18 laps (3 riders completing 6 laps), lap counting stops. The winning school would have done 18 laps, which means the maximum any other school could have done is 17. The schools are then sorted according to number of laps completed.

Add code to the Controller class to achieve this. You may create any additional helper methods. Call the appropriate method to display only the top three schools in your interface.

An example of output follows:

```
Midstream: 18
Waterkloof: 17
Menlopark: 17
```

[12]

80 marks

Total: 120 marks