NATIONAL SENIOR CERTIFICATE EXAMINATION
NOVEMBER 2015

INFORMATION TECHNOLOGY: PAPER II

Time: 3 hours 120 marks

PLEASE READ THE FOLLOWING INSTRUCTIONS CAREFULLY

1. This question paper consists of 13 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.

SCENARIO

The South African Amateur Tennis Players Association (SAATPA) wants to implement various software-based solutions in order to streamline their operation. More specifically, they want a database to keep track of the major grand-slam tournaments and a program to process the players and matches for an upcoming international tournament.

SECTION A STRUCTURED QUERY LANGUAGE

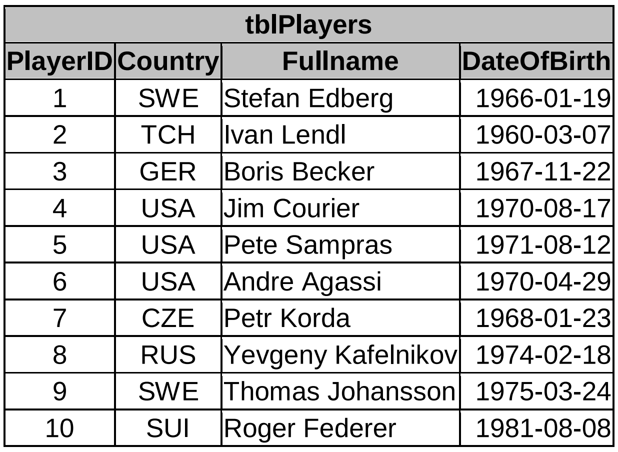
Professional tennis has four major grand-slam tournaments that are played on an annual basis. SAATPA wants to keep a record of all the winners of these four tournaments for the past few decades. They would like to record the details of each tournament, the details of the players, and the results of the final matches for each year of each tournament.

The database **TennisDB** contains three tables. The first table **tblPlayers** contains the details of each player including their name, the country they represent(ed) and their date of birth. The second table **tblTournaments** contains information on each of the four major tournaments, including the name of the tournament, the host city, the type of surface, the year the tournament was established and the prize money (in US dollars and South African rands). The prize money field for ZAR is intentionally left blank. The third table **tblFinals** contains details of the winner and runner-up tennis players who participated in the final of each tournament in each year, the year the final was held and the name of the final. You are required to extract some useful information from the database.

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

tblPlayers

Field Name	Data Type	Description
PlayerID	AutoNumber	This field assigns a unique ID for each player. This field is an autonumber field.
Country	Text	This field contains the player's country as text.
Fullname	Text	This field contains the player's full name as text.
DateOfBirth	Date/Time	This field contains the player's date of birth as a date/time.



tblPlayers			
PlayerID	Country	Fullname	DateOfBirth
1	SWE	Stefan Edberg	1966-01-19
2	TCH	Ivan Lendl	1960-03-07
3	GER	Boris Becker	1967-11-22
4	USA	Jim Courier	1970-08-17
5	USA	Pete Sampras	1971-08-12
6	USA	Andre Agassi	1970-04-29
7	CZE	Petr Korda	1968-01-23
8	RUS	Yevgeny Kafelnikov	1974-02-18
9	SWE	Thomas Johansson	1975-03-24
10	SUI	Roger Federer	1981-08-08

tblTournaments

Field Name	Data Type	Description
TournamentID	Autonumber	This field contains a unique ID for each tournament. This field is an autonumber field.
TournamentName	Text	This field contains the tournament name as text.
City	Text	This field contains the tournament's host city as text.
Surface	Text	This field contains the type of surface as text.
YearEstablished	Number	This field contains the year that the tournament was first held as number.
PrizeMoney	Number	This field contains the amount of prize money in US dollars as a real number.
PrizeMoneyZAR	Currency	This field contains the amount of prize money in South African rands. Deliberately left blank.

tblTournaments						
TournamentID	**TournamentName**	**City**	**Surface**	**YearEstablished**	**PrizeMoney**	**PrizeMoneyZAR**
1	Australian Open	Melbourne	Hard	1905	2457000	
2	French Open	Paris	Clay	1891	1873162	
3	Wimbledon	London	Grass	1877	2675200	
4	US Open	New York City	Hard	1881	1900000	

tblFinals

Field Name	Data Type	Description
TournamentID	Number	This field contains the tournament ID of the final match as a number. This field is a foreign key to tblTournaments.
FinalYear	Number	This field contains the year of the final as a number.
FinalName	Text	This field contains the type of final as text. Possible values as "Men's Singles" and "Women's Singles".
WinnerID	Number	This field contains the ID of the player who won the final match. This field is a foreign key to tblPlayers.
RunnerUpID	Number	This field contains the ID of the player who lost the final match. This field is a foreign key to tblPlayers.

tblFinals				
TournamentID	**FinalYear**	**FinalName**	**WinnerID**	**RunnerUpID**
1	1990	Men's Singles	2	1
2	1990	Men's Singles	30	6
2	1990	Women's Singles	17	16
3	1990	Women's Singles	50	16
3	1990	Men's Singles	5	6
4	1990	Women's Singles	15	96
4	1990	Men's Singles	1	3
1	1990	Women's Singles	16	70
3	1991	Women's Singles	17	15
4	1991	Men's Singles	53	3

QUESTION 1

1.1 Write a query that will list the **Country** and **FullName** of all **Players** sorted by country and then by name alphabetically. (3)

1.2 Write a query that will list all the details for any **Final** played prior to the year 2000. (3)

1.3 The player 'Bob Rogers' (**PlayerID 101**) is not a professional tennis player but was entered into the database in error. Write a query to delete his information from the **Players** table. (2)

1.4 Write a query that will update the **PrizeMoneyZAR** field in the **Tournaments** table with the equivalent South African rand amount (1 US dollar = 11.51 South African rand). (3)

1.5 Write a query that will show the total number of **Players** who have their birthdays in each month of the year. Show the month of the year as a number (January being 1 and December being 12) as well as the total number of birthdays in each month. The output should be as follows: (4)

BirthMonth	NumBirthdays
1	7
2	9
3	8
4	9
5	5
6	13
7	6
8	10
9	12
10	8
11	4
12	9

1.6 Write a query that will display the total number of **Players** that each country has in the database. Display only the countries that have more than 10 players. Display the **Country** as well as the total number of players in descending order. An example of the output is given below: (6)

Country	TotalPlayers
USA	14
ESP	11

1.7 Write a query to display the **TournamentName** of all **Tournaments** that have **PrizeMoney** greater than the average **PrizeMoney** of all the tournaments. An example of the output is given below: (4)

TournamentName
Australian Open
Wimbledon

1.8 Write a query that will list the **FullName**, **TournamentName** and **FinalYear** of all players who won a tournament in a leap year after 1995. A leap year is a year that is perfectly divisible by 4, but years that are perfectly divisible by 400 are not included. An example of the first 5 rows of output is given below: (8)

FullName	TournamentName	FinalYear
Monica Seles	Australian Open	1996
Boris Becker	Australian Open	1996
Yevgeny Kafelnikov	French Open	1996
Steffi Graf	French Open	1996
Steffi Graf	Wimbledon	1996

1.9 As often happens, the man and woman who won the Australian Open in 2015 also won the French Open against the same opponents. Write a query that will use the data in the **Finals** table to insert the same winners of the 2015 Australian Open (men and women's finals) for the 2015 French Open. The only inserted value that you may hardcode is the TournamentID. (7)

40 marks

SECTION B OBJECT-ORIENTED PROGRAMMING

SAATPA wants to host a South African Amateurs Open tennis tournament in South Africa. The first round of the men's tournament will consist of 32 international players. Each player will be given a seed and play against one other player in a match.

Each **Player** has the following information:
- Seed – the ranking of the player for the tournament
- Full Name – the name of the player
- Country – the country that the player represents

In the first round, each player will play one, and only one, other player, making a total of 16 matches. In order to make the tournament more exciting, the top seed player will play the bottom seed player. The player that is seeded second will play in a match against the player seeded second from the bottom and so on. In other words, Seed 1 plays Seed 32, Seed 2 plays Seed 31, Seed 3 plays Seed 30, etc.

Each **Match** contains the following information:
- A unique match code (explained below)
- The details of the two players in the match
- The final score/result of the match – given as the number of games won for each set
  - The score of a set is made up of the number of games each player won in that set. Player 1's score is listed first, followed by Player 2's score. (Eg 6-0 means that Player 1 won 6 games and Player 2 won no games.)
  - The score of a match is made up of the score for each set (eg 6-0, 6-8, 6-4, 6-3).

The **MatchCode** is the letter 'M' followed by the seed of the first player, and then by the seed of the second player. For example, in the match where Seed 4 plays Seed 29, the MatchCode will be 'M429'.

You have been given a file called **players.txt** that contains the information of exactly 32 players participating in the first round of the men's tournament. There will always be exactly 32 players in the given file. A sample of the first 10 lines of the file is given below:

```
1,Colby Noble,Hong Kong
2,Craig Hamilton,Myanmar
3,Dante Franklin,South Korea
4,Chadwick Baird,India
5,Wallace Forbes,Cameroon
6,Tanek Jimenez,Italy
7,Jared Flynn,Burundi
8,Yasir Rosario,Taiwan
9,Levi Klein,Botswana
10,Kibo Ford,American Samoa
```

Each line of this file contains the information of a single player in the men's tournament. The line is formatted as follows:

```
seed,name,country
```

QUESTION 2

Use the class diagram below to create a new class called **Player**. This class will be used to create objects that will store the details of one player in the tournament. The diagram below indicates the properties/fields and methods that are required.

Player
Properties/Fields: – Integer seed – String fullname – String country
Methods: + Constructor(Integer inSeed, String inFullName, String inCountry) + getCountryCode : String + toString() : String

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

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

2.3 Write code to create a constructor method that accepts an integer and two strings as parameters, which represent the seed, player's full name and country respectively. These parameters should be used to set the values of the three properties/fields of the **Player** class. (3)

2.4 Write code to create a method called **getCountryCode** that will return a String containing a three-letter code for the player's country. The three-letter code is always upper case and is generated as follows:

- If the name of the country is a single word, use the first three letters of the country name (eg 'China' becomes 'CHI').
- If the name of the country consists of two words, use the first letter of the first word and the first two letters of the second word (eg 'South Africa' becomes 'SAF'). (4)

2.5 Write code to create a **toString** method that will return the player's full name and country code in brackets, separated by a space. The format is as follows:

```
fullname<space>(country code)
```

for example:

```
Colby Noble (HKO)
```

(3)

[14]

QUESTION 3

Use the class diagram below to create a new class called **Match**. This class will be used to store the details of a single Match. Two of the properties/fields are **Player** objects (the class created in the previous question). The diagram below indicates the properties/fields and methods that are required.

Match
Properties/Fields: – String matchCode – **Player** player1 – **Player** player2 – String score
Methods: + Constructor(String inMatchCode, **Player** inPlayer1, **Player** inPlayer2) + getMatchCode : String + setScore(String inScore) + toString() : String

3.1 Write code to create a new class called **Match**. (1)

3.2 Write code to create four properties/fields that will store the **matchCode**, **player1**, **player2** and **score** associated with a match. Choose appropriate data types for these properties/fields, but note that the **player1** and **player2** properties/fields are objects of the class **Player**. (3)

3.3 Write code to create a constructor method that will initialise all the properties/fields of the **Match** class. Note that, in addition to the **matchCode** parameter, the second and third parameters are objects of the **Player** class. Use the first three parameters to assign values to their respective field. Set the score attribute to 'X' to indicate that the match does not yet have a score. (4)

3.4 Write code to create an accessor method for the **matchCode** property. This method should return a String. (1)

3.5 Write code to create a mutator method for the score property. This method should take in a single String parameter and set the value of the score field accordingly. (1)

3.6 Write code to create a **toString** method that will return a String comprised of the information for the match. The string returned should be in the following format:

```
Player1 vs. Player2 Score
```

If the score field is set to 'X', display 'Not Yet Played' for the score, otherwise just display the contents of the score field. For example, a match that has no score:

```
M132 Colby Noble (HKO) vs. Jason Underwood (POR) Not Yet Played
```

An example of a match that has a score is as follows:

```
M429 Chadwick Baird (IND) vs. Knox Olson (SOM) 6-0 6-8 6-4 6-3
Player 1 wins
```

(6)

[16]

QUESTION 4

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

4.2 Write code to declare the following two arrays as instance variables:
- An array that can be used to store exactly 32 **Player** objects.
- An array that can be used to store exactly 16 **Match** objects. (4)

4.3 Write code to create a constructor method that will read the contents of a file of players. The file name should be given as a String parameter to the constructor method. Each line will result in one **Player** object being added to the array. Do the following:
- Check if the file exists. If it does not, display an error message.
- Open the file for reading.
- Loop through the file 32 times. In each iteration of the loop:
  - Read in each line and split the **Player** data contained in that line into the seed, full name and country.
  - Create a **Player** object using the data.
  - Add the **Player** object to the array.
- Note that you do not need to create any **Match** objects in this method. (10)

4.4 Write code that will create a method called **listAllPlayers**. This method should return a string that contains the fullName and countryCode of all players. Use the object's **toString** methods that you created in the questions above. Each player's details should be on a new line. (5)

4.5 Write code to create a method called **populateMatches** that will generate 16 **Match** objects and add them to the array used to store matches. The first player in the player array (seed 1) will be the first player in the first match. The last player in the array (seed 32) will be the second player in the first match giving the matchCode 'M132'. The second seeded player will play the player with seed 31 in match 2 (matchCode 'M231'). Use the appropriate **Player** objects in the players array as parameters when creating a new match. Once you have populated the matches array with **Match** objects, your method should return a list of matches as a string. (7)

4.6 Write code to create a method called **findMatch** that takes in a matchCode as a String parameter and returns the corresponding **Match** object from the matches array. This method will search the matches array for the match with the same matchCode and return that object. You need not account for the case where a match is not found. (5)

[32]

QUESTION 5

5.1 Write code to create a simple user interface called **TennisUI** that will allow simple output. (1)

5.2 Declare and instantiate a **TournamentManager** object at the appropriate place in the code. Call the constructor using **'players.txt'** as the filename argument. (1)

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

Display all Players
Populate and Display Matches (2)

[4]

QUESTION 6

You are now required to write code to process the results of a match given in a text file in order to generate a score. In men's tennis tournaments, the scoring works as follows:

- Each match consists of at most 5 sets.
- The first player to win 3 sets wins the match.
- Each set consists of a number of games. In order to win a set, a player must meet both of the following conditions:
  - They must win at least 6 games.
  - They must win at least 2 more games than their opponent.
- For the purposes of this question, the points scored in individual games are not relevant.
- For the purposes of this question, assume that there is no tie-breaker rule.

You have been given a file called **results.txt**, which contains information about the results of a particular match. The first 10 lines of the file are given below:

```
M429
1
1
1
1
1
1
2
2
2
```

The first line of the file contains the **MatchCode** of the match that the results in the file relate to. Each line after the first line indicates which player won a particular game.

For example, in the sample given above, the results relate to match M429. The next lines indicate that Player 1 won the first six games and Player 2 won the next three games. In other words, lines 2 – 7 of the file are all games Player 1 won, which means he won the first set 6-0. The next three lines indicate that Player 2 won the next three games of the second set, and so on. The final score of match M429 is 6-0 6-8 6-4 6-3.

Below is a full explanation of how the match in the given file is to be processed. Note that your solution must cater for any results file. Hardcoding a solution that works only for the file given will be awarded little or no marks.

Line 1	M429	The Match Code
Lines 2 – 7 represent set 1	1 1 1 1 1 1	Player 1 wins 6 games and Player 2 wins 0 games. The final score for this set is 6-0.
Lines 8 – 21 represent set 2	2 2 2 1 1 1 2 1 2 1 1* 2 2 2	Player 1 wins 6 games and Player 2 wines 8 games. The final score for this set is 6-8. *At this point, the score is 6-5, but the set must be won by a difference of 2 games, so play continues until the final score is 6-8.
Lines 22 – 31 represent set 3	1 1 2 2 1 1 2 2 1 1	Player 1 wins 6 games and Player 2 wins 4 games. The final score for this set is 6-4.
Lines 32 – 40 represent set 4	1 1 1 2 2 1 1 2 1	Player 1 wins 6 games and Player 2 wins 3 games. The final score for this set is 6-3.

Player 1 wins the entire match as he wins 3 sets (sets 1, 3 and 4), and Player 2 only wins 1 set (set 2).

6.1 Write code to add a method in the **TournamentManager** class called **processResults**. This method should process the contents of **results.txt** and return the details of the match (including the score) as a string.

The format of the score is as follows:

- Each set is displayed as the number of games won by Player 1, followed by a dash, followed by the number of games won by Player 2. In other words, 6-0 indicates that Player 1 won 6 games and Player 2 won none. Player 1's score is always listed first.
- Each set is separated by a space, eg 6-0 6-8 6-4 6-3.
- Append the winning player at the end of the string, eg 6-0 6-8 6-4 6-3 Player 1 wins. (13)

6.2 Write code in **TennisUI** to call the processResults method and display the details of the affected match. Your output should be as follows:

```
Match Result : M429 Chadwick Baird (IND) vs. Knox Olson (SOM) 6-0
6-8 6-4 6-3 Player 1 wins
```

(1)

[14]

80 marks

Total: 120 marks