



eduresourceza.com



SUBJECT	Information Technology	DATE	July 2015
GRADE	12	MARKS	120
EXAMINER NAME	Mrs Kench	MODERATOR	Mrs Gill
USER NAME	Boake	DURATION	3 hours
PASSWOD	hsexam2		
	YJSUJC		

COGNITIVE LEVELS			
LOW ORDER	30 %	MIDDLE ORDER	40 %
		HIGH ORDER	30 %

STUDENT'S RESULT	
COMMENT	
TEACHER'S SIGNATURE	
PARENT'S SIGNATURE	

SECTION A: SQL**40 MARKS**

The files required for this question can be found in the folder named **Section A**. Please complete your work in this folder.

The United Nations Information Service has collected data about member countries in a database named **UNCountryStats**. The database's accuracy is questionable and requires some updating; nonetheless, you will be required to extract information from the database using SQL queries.

Table designs:**Country**

Field Name	Data Type	Description
Code	Text	A unique text code for each record.
CountryName	Text	The Country's name.
Area	Number	The total last recorded area of the country.
Population	Number	The total last recorded population of the country.
Census	Date/Time	The date on which the last census took place in the country.

Code	CountryName	Area	Population	Census
A	Austria	83850	8023244	2010/08/13
AXA	Anguilla	102	14436	2012/05/07
BI	Burundi	27830	5943057	2014/03/05
C	Cuba	110860	10951334	2011/04/12
COOK	Cook Islands	240	11870	2014/04/27
E	Spain	504750	39181114	2014/04/11
F	France	547030	58317450	2008/10/22
GB	United Kingdom	244820	58489975	2013/04/10
GR	Greece	131940	10538594	2011/12/16
I	Italy	301230	57460274	2013/01/10
K	Cambodia	181040	10861218	2011/10/19
LB	Liberia	111370	2109789	2012/12/27
MC	Monaco	1	31719	2014/06/02
MV	Maldives	300	270758	2013/01/25
NIUE	Niue	260	1398	2009/02/07
PE	Peru	1285220	24523408	2008/04/07
RB	Botswana	600370	1477630	2009/04/18
RL	Lebanon	10400	3776317	2013/05/28
RT	Togo	56790	4570530	2013/12/14

Province

Province		
Field Name	Data Type	Description
ID	AutoNumber	A unique number for each record.
ProvinceName	Text	The Province's name.
Country	Text	A foreign key that represents the Country to which the Province belongs.
Population	Number	The total last recorded population of the province.
Area	Number	The total last recorded area of the province.

Province					
ID	ProvinceName	Country	Population	Area	
1	Anatoliki Makedonia	GR	574308	14157	
2	Attiki	GR	3522769	3808	
3	Dhytiki Ellas	GR	702027	11350	
4	Dhytiki Makedonia	GR	292751	9451	
5	Ionioi Nisoí	GR	191003	2307	
6	Ipiros	GR	339210	9203	
7	Kedriki Makedonia	GR	1729581	19147	
8	Kriti	GR	537183	8336	
9	Notioi Aiyaior	GR	262522	5286	
10	Peloponnisos	GR	605663	15490	
11	Stereia Ellas	GR	578876	15549	
12	Thessalia	GR	731030	14037	
13	Voreioi Aiyaior	GR	189541	3836	
14	Alsace	F	1624000	8280	
15	Aquitaine	F	2796000	41309	
16	Auvergne	F	1321000	26013	
17	Basse Normandie	F	1391000	17589	
18	Bretagne	F	2796000	27209	
19	Bourgogne	F	1609000	31582	

Record: 1 of 214 No Filter Search

City

Note that some cities have missing values for latitude and longitude.

Field Name	Data Type	Description
ID	AutoNumber	A unique number for each record.
CityName	Text	The city's name.
Province	Number	A foreign key that represents the Province to which the City belongs.
Capital	Yes/No	A Boolean value that indicates whether the City is a capital city, or not.
Population	Number	The last recorded population of the city.
Longitude	Number	The Longitude co-ordinate of the city.
Latitude	Number	The Latitude co-ordinate of the city.

ID	CityName	Province	Capital	Population	Longitude	Latitude
1	Tameside	159	False	221800		
2	Oldham	160	False	220400		
3	Trafford	161	False	218100		
4	Rochdale	162	False	207100		
5	Bury	163	False	181400		
6	Winchester	151	False	100500		
7	Southampton	122	False	211700		
8	Portsmouth	129	False	189100		
9	Basingstoke	128	False	146500		
10	Havant	127	False	117500		
11	Eastleigh	126	False	109600		
12	Fareham	125	False	101000		
13	Worcester	130	False			
14	Saint Albans	123	False	127700		
15	Hertford	118	False			
16	Kingston upon	121	False	269100		
17	Beverley	120	False	115800		
18	Hull	119	False			
19	Maidstone	139	False	138500		

QUESTION 1

40 MARKS

- 1.1 The UN Information Service wants a list of all the country names and their populations, sorted in descending order of population and then alphabetically according to the country name. (4)
- 1.2 The Information Service wants a list of city names of all cities that have a **Population** greater than 100000, are **Capital** cities and are in the **Latitude** range of -20 to 20 (inclusive). Order the list alphabetically based on **CityName** field. Note that some cities have missing values for latitude and longitude. (4)
- 1.3 A new city 'Dudley Heights' has been proclaimed as its population has grown to 100001. Perform a SQL statement to add the new city with the exact details of 'Colchester'. (5)
- 1.4 In the table **City**, there are many blank entries for **Latitude** and **Longitude**. Some applications may find it difficult to work with values of this type so it has been decided to give a default value of 0.1 to all of the blank values. Perform a SQL query to update all of the blank values. Note: values that are not blank must remain unchanged. (4)
- 1.5 Provide a list of the countries and how many years have elapsed since each one's last **Census**. (3)

- 1.6 List all of the countries that have more than 1 city recorded in the database. Show the **CountryName** and **Number of Cities** as fields in the query. (7)
- 1.7 The data in the database is suspected of being quite inaccurate. In order to check this, devise a query that will display the percentage variance between the **Area** of a country and the total **Area** of the provinces in each country. The query must display the fields **CountryName**, **Country Area**, **Total Provinces Area** (for each **Country**) and the percentage **Variance**. The percentage variance is calculated by the following:
- $$\text{Variance} = (\text{Country Area} - \text{Total Provinces Area}) / \text{Country Area} \times 100.$$
- The **Variance** field must be displayed correct to 2 decimal places. (8)
- 1.8 Provide a list of province names and province populations for the provinces that have a **Population** which is *lower* than the average **Population** of all provinces. The list must only include the 3 highest populations, in descending order. (5)

SECTION B

OOP

80 MARKS

The national lottery organisers require a simple, yet efficient, application that will issue a random sequence of 6 numbers (in the range 1 to 49) for each ticket sold. The ticket is generated at the sales point with a date and time; a request is then sent to the 'Lotto' URL to generate and return the random number sequence for the ticket. There are many Lotto tickets. Your task is to provide a simulation of the process and address some statistical outputs.

You have been provided with the file **LottoTickets.txt** in the **Section B** folder. The file contains a collection of dates and times (separated by '#' symbols), as generated by a sample sales point over a week of sales. The example below shows the 1st 11 lines of the sample:

Date#Time

```
28/06/2015#08:23
28/06/2015#08:30
28/06/2015#08:31
28/06/2015#08:39
28/06/2015#08:56
28/06/2015#09:08
28/06/2015#09:13
28/06/2015#09:44
28/06/2015#10:12
28/06/2015#11:05
28/06/2015#11:20
```

QUESTION 2

28 MARKS

- 2.1 Create a class called **Numbers** with an array of 6 numbers. The array must only be directly visible within this class. (3)
- 2.2 Code a constructor that will instantiate the elements of the array with 6 numbers that are randomly generated between 1 and 49. (4)
- 2.3 Code a **toString** method that will return a string consisting of the elements of the array as a single string with a space separating the elements. For example, if the array consists of elements (49,32,21,5,44,17), the **toString** method must return the string '49 32 21 5 44 17'. (4)
- 2.4 Code a **sort** method that will sort the array of 6 numbers from lowest to highest. (6)

- 2.5 Add code to the constructor to ensure that the six numbers are unique with no duplicates. (6)
- 2.6 Code a method called **countNumbers** that will receive a string of 7 numbers and determine how many numbers in the array are in the string. The method must return an integer value with the count of the number of numbers in the array that are in the string parameter. The string parameter should have the same format as that of the **toString**, i.e. '1 2 3 4 5 6 7', with spaces between the numbers. (5)
- This method will be needed in Question 6.

QUESTION 3

19 MARKS

- 3.1 Code a class called **Ticket** that will contain 2 string instance variables for the date and time of the issued ticket. These variables must only be directly visible within this class. Another object instance variable of the **Numbers** class must be included without visibility outside of this class. (4)
- 3.2 Code a parameterized constructor for this class that will receive values for the date and time instance variables and instantiate the **Numbers** object instance variable. (4)
- 3.3 Code a typed method called **changeTime** to represent the 24 hour time as 12 hour time. In other words if the hour is 13, which is after 12, then subtract 12 and add pm to change the time from 13:08 to 1:08pm. The method must return the 12 hour time as a string. Morning times must be represented in the same format with the characters 'am' after the time value. (6)
- 3.4 Write a **toString** method that will return the date, and the time in 12 hour time using the method above. After the time, display the **sorted** numbers on the same line. according to the following format:
- If the date is '28/06/2015', time is '3:23pm' and the **toString** method of the **Numbers** object returns '5 17 21 32 44 49', the result must be:
- ```
'>>>28/06/2015>>>3:23pm***5 17 21 32 44 49'
```
- (5)

### QUESTION 4

15 MARKS

- 4.1 Code a class called **TicketArray** to store 200 tickets and a variable to record the number of **Ticket** objects in the array. (3)
- 4.2 Code a constructor to instantiate the array of tickets using the text file **LottoTickets.txt**. This constructor must send 2 parameters to the constructor of the **Ticket** object (Date and Time). (8)
- 4.3 Code a **toString** method to display each ticket, in the format of the **toString** of the **Ticket** element. Each ticket element must appear on a new line with a blank line in between. (4)

### QUESTION 5

3 MARKS

- 5.1 Add code to instantiate a **TicketArray** object. (2)
- 5.2 Add code to display all the tickets by calling the **toString** method of the **TicketArray** object. (1)

**QUESTION 6****15 MARKS**

The Lotto organisers would like to determine the lotto tickets with the most number of matching numbers. The user will enter 7 numbers (six numbers and the bonus ball in any order) and the ticket with the most number of matching numbers must be displayed. If there are more than one ticket with the same high number of matching numbers then all tickets with the same high number of matching numbers must be displayed.

The program must be adapted to input the seven numbers as a single string separated by spaces, call any relevant methods and then display the result. (15)

---

eduresourceza.com

eduresourceza.com