**Challenge 1 - FizzBuzz**

Add Unit Tests that prove the logic of your methods.

**Step 1:**

Write some code that prints out the following for a contiguous range of numbers:

\* the number

\* 'fizz' for numbers that are multiples of 3

\* 'buzz' for numbers that are multiples of 5

\* 'fizzbuzz' for numbers that are multiples of 15

e.g. if I run the program over a range from 1-20 I should get the following output:

*1 2 fizz 4 buzz fizz 7 8 fizz buzz 11 fizz 13 14 fizzbuzz 16 17 fizz 19 buzz*

Archive this as a separate zip file then continue on to step two.

**Step 2:**

Enhance your existing FizzBuzz solution to perform the following:

\* If the number contains a three you must output the text 'lucky'. This overrides any existing behaviour.

e.g. if I run the program over a range from 1-20 I should get the following output:

*1 2 lucky 4 buzz fizz 7 8 fizz buzz 11 fizz lucky 14 fizzbuzz 16 17 fizz 19 buzz*

Archive this as a separate zip file then continue on to step three

**Step 3:**

Enhance your existing solution to perform the following:

\* Produce a report at the end of the program showing how many times the following were output:

\*\* fizz

\*\* buzz

\*\* fizzbuzz

\*\* lucky

\*\* an integer

e.g. if I run the program over a range from 1-20 I should get the following output:

*1 2 lucky 4 buzz fizz 7 8 fizz buzz 11 fizz lucky 14 fizzbuzz 16 17 fizz 19 buzz*

*fizz: 4*

*buzz: 3*

*fizzbuzz: 1*

*lucky: 2*

*integer: 10*

(Integer is 10 because there were 10 numbers that were not altered in any way).

Archive this as a separate zip file.

**Challenge 2 - GetWordFromText Method**

Implement a method *GetWordFromText(string text, int position)* that returns a specific word of an input text, given the specified position. Words are separated by space (' ').

For example:

GetWordFromText("one two three", 2) should return "two"

GetWordFromText("one;two three", 2) should return "three"

GetWordFromText("one", 1) should return "one"

Be aware that:

When input parameter *position* is less than 1, the method should throw an Exception.

When input text does not have enough words (GetWordFromText("one", 2)), method should throw an Exception.

The method should ignore all spaces at the beginning and at the end of the input text.

Add any other checks you think can be helpful to improve the program.

Add Unit Tests that prove the logic of your methods.

**Challenge 3 - Reverse Method**

Implement a method that returns a reversed string of a given input value.

For example:

Reverse(“one”) should return *eno*.

Reverse(“abcd dcba”) returns *abcd dcba.*

Add any checks you think can be helpful to improve the program.

Add Unit Tests that prove the logic of your methods.

**Challenge 4 - SQL**

**Create the database with the following script and write the queries for steps 1 and 2**

*CREATE DATABASE KIGO;*

*GO*

*USE KIGO;*

*GO*

*CREATE TABLE CATEGORIES*

*(*

*CATEGORY\_ID INT NOT NULL IDENTITY,*

*CATEGORY\_NAME VARCHAR(50) NOT NULL,*

*PRIMARY KEY(CATEGORY\_ID)*

*);*

*INSERT INTO CATEGORIES (CATEGORY\_NAME) VALUES*

*('Sports'),*

*('Actualités'),*

*('Animaux'),*

*('Economie'),*

*('Cuisine');*

*CREATE TABLE ITEMS*

*(*

*ITEM\_ID INT NOT NULL IDENTITY,*

*CATEGORY\_ID INT NOT NULL,*

*ITEM\_NAME VARCHAR(50) NOT NULL,*

*ITEM\_PRICE DECIMAL(8, 2) NOT NULL,*

*PRIMARY KEY(ITEM\_ID),*

*FOREIGN KEY (CATEGORY\_ID) REFERENCES CATEGORIES(CATEGORY\_ID)*

*);*

*INSERT INTO ITEMS (CATEGORY\_ID, ITEM\_NAME, ITEM\_PRICE) VALUES*

*(1, 'L''équipe' , 6.50),*

*(2, 'Le Monde' , 3.50),*

*(2, 'Le Parisien' , 2.50),*

*(2, 'France soir' , 3.00),*

*(3, '30 Million d''amis' , 6.20),*

*(3, 'Cheval pratique' , 4.50),*

*(4, 'Capital' , 2.50);*

**Step 1**

Write a SELECT that returns all categories sorted by (A) the number of items they have, and (B) the category name.

This query should fetch the following columns: the category name, the number of items (AS N\_ITEMS) and the average price of the titles in that category (AS AVERAGE\_PRICE)

**Step 2**

Write a SELECT query returning the CATEGORIES rows that don't have any associated item, or just 1 associated item.

Only the category name column needs to be fetched by this query.