XQUERY EXERCISE

For your XQuery exercises we'll be working with the same XML file we used yesterday. Because the XML elements in this collection are coded in the TEI namespace, we need to begin by declaring that TEI is our default element namespace (otherwise we will be unable to access the element nodes in the collection).

What do we use?

You can use Atom and then process with Xquilla on your command line.

If you are working with eXistDB, open eXide, and a new XQuery window, and paste in the following line, all the way to the semicolon, to establish that we are working in the TEI namespace:

```
declare namespace tei="http://www.tei-c.org/ns/1.0";
```

Query uses XPath expressions to find its way through its index of files. It can work on one file, or on a whole collection, thus:

The doc() function in XQuery finds a single document, and inside the parentheses goes the path to the document within the database, including the filename. To retrieve our example.xml use:

```
doc('/example.xml')
```

FLWOR Expressions in XQuery

Flower or FLWOR expressions are a powerful tool in XQuery, letting us work in more complex ways with querying and remixing information in files and collections—sometimes both in the same expression! Here's a primer on FLWOR (or really, LFWOR!):

- **Let**: establishes variables which may be single values or arrays of multiple values (single or multiple)
- **For**: establishes a **range variable** that moves step by step from one value to the next and the next in a long list of values defined by a Let statement. The range variable is designed to process and return a single value at a time, and we say that it *loops through* a list of multiple values.
- **Where** (optional): filtering; analogous to predicates
- **Order by** (optional): alphabatize, etc. If you use it with a Where statement, Order by always has to appear after Where. Always appears after a Where.

- **Return**: generates output

LET

1 - exercise

1. Write a script that retrieves the <add> element.
2. Write a script that *count* the number of <add> elements with the count function from Xpath
3. Use the { } to print the value - {count($add)}
4. Add the HTML element in the return value

The output: 13

1. Count the number of elements <add> the attribute: @hand="#PBS" (square brackets are use for a condition[…]
2.

The output: 8

2 - exercise

1. Write a script that retrieves the <title> and the <licence> elements
2. Add in the return:

```
<html>
  <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
        <h1>Manuscript of Frankenstein: TEI Transcription Frankenstein</h1>
        <p>CC-BY-SA
        4.0</p>
  </body>
</html>
```

To return the value use data: ex. data($variable)

The output:

```
<html>
  <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
        <h1>Manuscript of Frankenstein: TEI Transcription Frankenstein</h1>
```

```
        <p>CC-BY-SA
        4.0</p>
  </body>
</html>
```

FOR

1. Print the content of each <del> wrapped in a <p> HTML tags
2. Add the string-length function and the Xquery order by expression

Order by

1. Order by string-length
2. Add the condition to retrieve only the element <del> with the attribute type=”crossedOut”
3. Add <p> in the return

The output:

```
<p>h</p>
<p>And</p>
<p>dun</p>
<p>fomed</p>
<p>But how</p>
<p>the rain</p>
<p>handsome</p>
<p>handsome</p>
<p>Handsome</p>
<p>and endeavour to</p>
<p>the frame on whic</p>
```

Where

Add *where* and the xpath function *string-length* to filter by length the sentences deleted (del) to retrieve only what is longer than 8 characters.

1. When string-length is > 8

- Exercise:

ADD **If** - Conditions

1. Add in in the return block an if-stmt to check when the object has a “type” attribute set to “overwritten”.

```
if ($title/@type/data() = "overwritten")
        then <p class="overwritten">{$title/data()}</p>
        else <p class="non-overwritten">{$title/data()}</p>
```

The output:

```
<?xml version="1.0" encoding="UTF-8"?>
<p class="overwritten">at</p>
<p class="non-overwritten">And</p>
<p class="non-overwritten">dun</p>
<p class="non-overwritten">fomed</p>
<p class="non-overwritten">But how</p>
<p class="non-overwritten">the rain</p>
<p class="non-overwritten">handsome</p>
<p class="non-overwritten">handsome</p>
<p class="non-overwritten">Handsome</p>
<p class="non-overwritten">and endeavour to</p>
<p class="non-overwritten">the frame on whic</p>
```