**Introduction**

This exercise will cover **XQuery**. You will be using **oXygen** (https://www.oxygenxml.com/xml_editor/software_archive_editor.html), AN XML/JSON development IDE, in order to do this exercise. You should have received a license key for it by mail. Before starting, make sure oXygen is installed and working on your computer.

Download the XML/XSD files book_catalogue.xml (https://drive.google.com/uc?export=download&id=0B_c6n15v_sC7MmlscmFDeWlkDw), airport-flight-passenger.xm (https://drive.google.com/uc?export=download&id=0B_c6n15v_sC7Nlh2ZG51WWNJbl), airport-flight-p (https://drive.google.com/uc?export=download&id=0B_c6n15v_sC7WjIfVzdtaHFpeFk) in order to complete the exercise.

**1. XQuery Basics**

**FLWOR**

- **For** - Iterates over a sequence of nodes.
- **Let** - Binds a sequence to a variable.
- **Where** - Filters the nodes / specifies criteria for the result.
- **Order by** - Specifies sort order.
- **Return** - Specifies what need to be returned.

**Questions**

Run the following queries on book_catalogue.xml using oXygen:

1. Find book titles ordered lexicographically where book price is greater than 15.
2. Find books written in English which belong to category "CHILDREN" ordered by the year of publication.
3. Find books ordered by categories which are published after 2010 and costs less than 10.
4. Find all authors who have written books in the "BUSINESS" category before 2014 and sort their names lexicographically.
Conditional Expressions

If-Then-Else expressions are allowed in XQuery.

Question
Run the following query on book_catalogue.xml using oXygen:

1. List all the book titles. If the book belongs to category CHILDREN, put the title within <child></child> tag. Else put the title within <adult></adult> tag.

2. Namespaces in XQuery

Consider the following document:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE eth SYSTEM "eth.dtd">
<eth xmlns="http://www.ethz.ch"
     xmlns:db="http://www.dbis.ethz.ch"
     date="11.11.2006"
     db:date="12.11.2006">
  <date>13.11.2006</date>
  <president number="1">Empty</president>
  <Rektor>Name 2</Rektor>
</eth>
```

Question
Correct the following XQuery expressions so that they refer to the correct namespace, in order to determine the value of the date sub-element and of the two date attributes:

1. `/eth/date`
2. `/eth/@date`

Hint: you can bind a prefix pre to a URI http://www.example.com with the following prolog declaration:

```xml
declare namespace pre = "http://www.example.com";
```

You can then use the prefix in path expressions.
3. XQuery Comparison Operators

**General comparison operators:** =, !=, <, <=, >, >=

**Value comparison operators:** eq, ne, lt, le, gt, ge

The value comparison operators (eq, lt, etc.) are designed to compare single values (i.e. sequences of one value each). The "general comparison" operators (=, <, etc.) are designed to compare sequences of more than one values.

"foo" eq ("foo","bar") throws an error.
"foo" = ("foo","bar") returns true. With a general comparison operator, the expression will return true if any of the items on the left compare successfully with any of the items on the right. This is sometimes called "existential quantification."

**Questions**

1. Find a variable binding for $x$ so that $x=1$ and $x=2$. Can one infer that, in XQuery, $1=2$?
2. Find variable bindings for $x$, $y$ and $z$ so that $x > y$ and $y > z$, but $x > z$ is not true.
3. Find a variable binding for $x$ so that neither $x$ eq $x$ nor $x =$ $x$ is true. Explain why.

4. Atomization and Effective Boolean Value

**Atomization**

Atomization is the process of extracting the typed value of an item. This process is implied under certain circumstances. Some of the XQuery operators, such as arithmetic and comparison operators, depend on this process. For example, when you apply arithmetic operators directly to nodes, the typed value of a node is first retrieved by implicitly invoking the data function. This passes the atomic value as an operand to the arithmetic operator.

**Question**

Try to guess the outcome of the following XQuery expression:

```xquery
fn:sum(
    for $x in doc("book_catalogue.xml")//book
    where $x/@category eq "BUSINESS"
    return fn:sum($x/price)
)
```
**Effective Boolean Value**

The effective Boolean value (EBV) of a sequence is computed implicitly during the processing of expressions that require Boolean values.

**Question**

Find the EBV (true/false) for the following descriptions of values:

1. An empty sequence
2. A sequence whose first item is a node
3. A single value of type xs:boolean
4. A single value of type xs:string
5. A single value of any numeric type

**5. XQuery Functions**

XQuery is built on XPath expressions. XQuery 1.0 and XPath 2.0 share the same data model and support the same functions and operators. Here is the [link](https://www.w3.org/TR/xpath-functions/) to the documentation.

A call to a function can appear where an expression may appear.

**Examples:**

In an element:

```xml
for $x in doc("book_catalogue.xml")/bookstore/book
where $x/@category="CHILDREN"
return <name>{upper-case($x/title)}</name>
```

The above XQuery expression returns all books in the CHILDREN category in upper case.

In the predicate of a path expression:

```xml
for $x in doc("book_catalogue.xml")/bookstore/book[substring(title,1,4)='Alg o']
return $x/title
```

The above XQuery expression returns title elements for the books whose titles begin with Algo.

If you cannot find the XQuery function you need, you can write your own:
Declare function prefix:function_name($parameter as datatype) as returnDatatype { ...function code here... };

- Use the declare function keyword
- The name of the function must be prefixed
- The data type of the parameters are mostly the same as the data types defined in XML Schema
- The body of the function must be surrounded by curly braces

**Question** Consider that the prices of books in the book_catalogue.xml file are given in USD. Find the price of the book titled ‘If Animals Kissed Good Night’ in Euros (consider 1USD = 0.92EUR). Write a function that takes as input the price in USD and gives as output the price in EUR.

### 6. More XQuery Expressions

Consider the XML document airport-flight-passenger.xml validated against airport-flight-passenger.xsd distributed together with this exercise (download links in Introduction section). Write the XQuery expressions for solving the following problems. Your results should be well-formed XML.

1. Give the list of direct flights on 2009-12-24 which have "North Pole" (airport name) as the source airport.
2. Retrieve the busiest airport on 2009-12-24 (based on the number of departures and arrivals).
3. Identify all the flight destinations of Passenger "Santa Claus".
4. Consider the case of combined flights (one or more intermediate stops). As an example, flying from London to Zurich on 2008-12-24 might mean taking two separate flights: London-Amsterdam and Amsterdam-Zurich, both on the same date. Retrieve all flight possibilities from "North pole" to "South pole" on 2009-12-24 with one or two intermediate stops.

### 7. Analogy with SQL

Have you noticed that the XQuery FLWOR expressions are similar to SQL SELECT statements? Of course, the **for** expression in XQuery identifies XML nodes while the **FROM** clause in SQL identifies tables from which rows are chosen. But this does not restrict us from trying to map the sub-expressions of XQuery FLWOR with the analogous syntax elements of SQL's select.

**Question**
Find the SQL syntax elements for the following XQuery expressions: for, where, order by, return