0. List the entire contents of Mondial.
   
   \[ \text{doc("mondial.xml")/mondial} \]

1. Give a list of all the countries in XML.

   \[
   \text{<result>}
   \{ \text{doc("mondial.xml")//country} \}
   \text{</result>}
   \]

2. Give a list of the countries that Germany borders.

   \[
   \text{<result>}
   \{ \text{doc("mondial.xml")//country[@car_code="D"]/border} \}
   \text{</result>}
   \]

   To get the names:
   \[
   \text{<result>}
   \{ 
   \text{for } \$x \text{ in doc("mondial.xml")//country[border[@country="D"]]/name}
   \text{ return } \$x
   \}
   \text{</result>}
   \]

3. Give the names of all the countries with population at least 10 million.

   \[
   \text{<result>}
   \{ \text{doc("mondial.xml")//country[population/text() >= 10000000]/name} \}
   \text{</result>}
   \]

   To do the comparison, you need to obtain the character string within each `<population>` element. You do this by using the `text()` function of XPath as an immediate "child" of population/. XPath will then coerce the string to a number automatically.

   Another way to write this query:
   \[
   \text{<result>}
   \{ \text{doc("mondial.xml")//country[population >= 10000000]/name} \}
   \text{</result>}
   \]

   (If a element has only text, its name can be used without having to specifically use the `text()` function of XPath.)

4. Find all cities located in countries that are partially or fully part
of Europe. (The cities themselves don't have to be in Europe.)

```xml
<result> {
    doc("mondial.xml")//country[encompassed/@continent="europe"]//city
} </result>
```

Conditional expressions can have complex XPath expressions inside as well. Here we search for countries by matching an attribute of a `<country>`'s subelement.

5. Find the names of all rivers that start north of the equator (at a positive latitude).

```xml
<result> {
    doc("mondial.xml")//river[source/latitude > 0.0]/name
} </result>
```

6. Find the names of all rivers that start in Iceland.

```xml
<result> {
    doc("mondial.xml")//river[source/@country = (//country[name='Iceland']/@car_code)]/name
} </result>
```

Notice how we have nested one absolute XPath expression inside another – we compare the country attribute against the ID code of the country named Iceland.

7. Get the names of all countries in both Asia and Europe.

```xml
<result> {
    doc("mondial.xml")//country[encompassed/@continent='europe' and encompassed/@continent='asia']/name
} </result>
```

How does this work?

In XPath, equality comparisons have implicit existential quantifiers. This means they return true if *one* of the items in the left-hand sequence matches *one* of the items in the right-hand sequence (either sequence can consist of just one item, such as 'europe' above). This is true of all comparison operators, actually.
Hence, since there exists an <encompassed continent='europe' /> subelement,
and another distinct <encompassed continent='asia' /> subelement,
there can be a match.

This would not work:

<result> { doc("mondial.xml")//country/encompassed[@continent='europe' and 
@continent='asia']/name 
} </result>

because here there really is only one item on each side of the equality
test, there being only one "continent" attribute in an
<encompassed>.

Alternative way:

<result> { doc("mondial.xml")//country[encompassed/@continent='europe'] 
[encompassed/@continent='asia']/name 
} </result>

XPath condition brackets stack from left to right.

8. Challenge problem:
Get the name of every country that borders France *and* has either
population greater than 20 million *or* GDP greater than 10000.

<result> { doc("mondial.xml")//country[border/@country = (// 
country[name='France']/@car_code)][population > 20000000 or gdp_total 
> 100000]/name 
} </result>