CS 345B Homework 2: Solutions

Instructors: Dr. Donald Kossmann and Dr. Daniela Florescu

Exercise 1:

Using the XML Document below, with the URI „bib.xml“ (library with books), define the following queries in XQuery:

a.) Give the titles of all Books sorted by Price.

```
for $b in doc("bib.xml")//book
order by xs:float($b/price) descending, $b/title ascending
return $b/title
```

Note: In the case when no cast to xs:float is used, the results are ordered alphabetically by price (price are considered strings). This means that ‘129.95‘ < ’39.95‘.

b.) How many books were written by Abiteboul?

```
doc("bib.xml")//book[exists(index-of(author, 'Abiteboul'))]
```

Alternative solution:

```
doc("bib.xml")//book[author =  'Abiteboul']
```

c.) Give for each author, the number of books he has written.

```
for $a in distinct-values(doc("bib.xml")//author)
return <res>
        <name>{$a}</name>
        <count>
        {count(doc("bib.xml")//book[exists(index-
of(author,$a))]) }
        </count>
       </res>
```

Exercise 2: Surprising XQuery

1. Prove that in XQuery 1=2

In XQuery the expression 1=2 will never evaluate as true. The two parts of the equality are promoted to the same type (numeric type) and in this case 1 can never be equal to two.

This exercise wanted however to make you aware of the way XQuery uses "=" when sequences are involved. In particular, the following hold:
1 = (1,2)
And
2 = (1,2). However, this just "might" let you think that 1=2. This is **false**, since "=" is not transitive.

2. Prove that if $x > $y and $y > $z, $x > $z is not necessarily true

Example:
$x := 2
$y := (1,4)
$z := 3

Exercise 3:

Consider an XML document corresponding to the model created for Exercise 1 (Exercise Sheet1).Write the XQuery expression for solving the folowing problems:

1. **Give the list of the direct flights on the date of 2005-12-24 which have "North Pole" (airport name) as the source airport.**

```
<flights>
{
for $a in //Airport[name/text() eq 'North Pole']
    return //Flight[(source eq $a/@airId) and (date/text() eq '2005-12-
24')]
}
</flights>
```

2. **Retrieve the list of the busiest airports on the date of '2005-12-24' (based on the number of departures and arrivals).**

```
let $results := (<order>
{
for $a in //Airport
let $c := count(//Flight[((./source/text() eq $a/@airId) or
(./destination/text() eq $a/@airId))] )
order by $c descending, $a/name ascending
return
    <result>
        {$a}
        <count>{$c}</count>
    </result>
}
</order>)
```

```
return $results/result[xs:integer(./count/text()) eq
xs:integer(max($results//count/text()))]
```

The idea of the solution is to:

- first compute a list of airports with their ‚load count' (number of arrivals and departures for that day). (let $results := ....)
- second: retreive from the obtained result those who have a value for count which is equals to maximum value of count (return $results[... max ($results//count./text...)...]).

2.a) TO MAKE THIS MORE COMPLICATED, LET'S COMPUTE HOW BUSY AN AIRPORT IS BASED ON THE NUMBER OF PASSENGERS USING THE AIRPORT IN THAT DAY:

(: Return the number of airports ordered by the **number of passengers** which fly through it (source or destination) on the day of '2006-12-24' :)

```
 (: compute for each passenger the list airports he is flyng in that day:)
 let $passengerVisits :=
 for  $p in //Passenger
     let $FlightIds := //Reservation[(passRef/text() eq $p/passportnumber/text()) and
date="2005-12-24"]/flightRef  (:  the passenger's reservations for the day :)
     let $Flights := for  $f in $FlightIds return //Flight[@flightId = $f]
     let $allVisitedAirports := $Flights/source/text() union $Flights/destination/text()
(:  all airports visited by the passenger :)
     return <visit>
               {$p}
               {for  $a in $allVisitedAirports
                   return  <airp>{$a} </airp>}
             </visit>

 (:  now we have a list of passengers, and for each we have a list of aiports. Now for
each of those airports, we want to see how many times it appears in the list, for any
passenger. :)
 for  $a in distinct-values($passengerVisits//airp)
     let $numberOfPassengers := count($passengerVisits//airp[$a = . ])
     order by $numberOfPassengers descending
     return <airport>
              <name>{$a}</name>
              <count>{$numberOfPassengers} </count>
            </airport>
```

The solution works because here are the flights which have a reservation on that specific day, and the corresponding airports (source and destination). The number of passengers for each aiport is the count of appeareances of each airport in this results:

Passenger 000111 → LX123: LHR ZRH
Passenger 000112 → LX140: FFT NPL
Passenger 000138 → LX138: FFT PRG
Passenger 000114 → LX168: AMS SPL

=> FFT (2), ZRH(1), NPL(1), PRG(1), AMS(1), SPL(1)

3. Identify all the flight destinations of the Passenger ‚Santa Claus'.

```
for $p in //Passenger[name eq 'Santa Claus'], $r in //Reservation
where $r/passRef eq $p/passportnumber
return distinct-values(//Airport[@airId eq //Flight[$r/flightRef eq
@flightId]/destination]/name)
```

Notes: The solution is based on a JOIN between the Passengers and the Reservations, which binds a separate variable ($p and $r) to each type of element.
The join is based on the condition:
$r/passRef eq $p/passportnumber
and retrieves all Reservations by the Passenger with the name ‚Santa Claus'. (in the reservation we have the flightId and we can retrieve the)

The final condition in the return clause:

- retrieves the flight destination (airport id) corresponding to the flightId in the Reservations
- retrieves the name of that specific aiport
