Coursework submission 2
Individual submission by Stian A. Johansen
XML and structured documents
deadline 10 December 2012

The XQueries are separated into three different files for them to be more readable.I use nested loops to iterate through the document a number of times. When a certain requirement is met (i.e the loops are positioned so that cousins are available as elements in the current position) the elements are returned. Before returning I also make sure they are not the same elements. All output validates directly as HTML5.

FirstCousin
From top to bottom, I use one for-loop for each "level" in the family tree. The first loop stores the left and right child (if-exists) of the family, the next loop goes left and the loop after that goes right in the tree (not literally, but I use them to check for children in the left and right branches of the tree). In the innermost of the loops I "lock" the iterations on the position where element i in the innermost iteration is related to the surrounding iteration in that i's father or mother is a child of the outer iteration. I do this twice for both sides of the tree. In addition I compare the bottom-most children to see if they are identical and store these three tests in variables. the where statements basically makes sure that the three conditions are true, then I return a piece of html where the left and right cousin(s) are in separate divs, printing related cousins after one another (if one person does not have a cousin, he does not show up at all). I use string-join to separate children of the same family by comma.

SecondCousin
This XQuery differs from FirstCousin in that it has two more for loops (because we have to iterate two more times to check the relation downwards). In addition of checking what I did for firstcousin, this time I check if theres a level below that as well, i.e checkin the level if its father/mother is a child of the lowermost valid iteration of firstcousin. If it is two levels deep on both sides, the children of those families are second cousins.

SecondCousinOnceRemoved
This XQuery is similar to secondCousin, but it uses one more iteration (because one member of the cousin pair is one level lower than on secondcousin). In addition, I save two boolean variables stating wether the lowest level is on the left or the right side. The where statement requires these to be different (i.e one of them is one level lower than the other, one true and one false), and then prints out the cousins using if statements on the aforementioned variables in the return statement.

Improvements
This algorithm is written on basis of the wikipedia examples, and has one serious disadvantage that it cannot handle comparisons in families with more than two children (it will just compare the first two children). This is because of the setup of my loops. To solve this I would have iterated on the basis of children instead of on the basis of the root document. By doing that I realise I could have had a much more efficient algorithm as well, because I could navigate from the child elements using xpath (navigating up to parent or down to child to do comparisons).

Files: family.dtd (identical to the one in coursework2.pdf only with comments), family1.xml, family2.xml, FirstCousin.xquery, SecondCousin.xquery, SecondCousinOnceRemoved.xquery, this document, fam1.JPG, fam2.JPG (illustrations of the two family-documents)