Problem 1 (20 points)

Consider XML documents conforming to the following DTD:

```xml
<!DOCTYPE Univ [    <!ELEMENT University (Dept+)>
    <!ATTLIST Dept Name ID #REQUIRED>
    <!ELEMENT Dept (Faculty | Staff | Student)>
    <!ATTLIST Faculty Name ID #REQUIRED>
    <!ELEMENT Faculty (Office, Salary)>
    <!ATTLIST Staff Name ID #REQUIRED>
    <!ELEMENT Staff (Office, Salary)>
    <!ATTLIST Student Name ID #REQUIRED
      Advisor IDREF #REQUIRED>
    <!ELEMENT Student (Dorm)>
    <!ELEMENT Office (#PCDATA)>
    <!ELEMENT Salary (#PCDATA)>
    <!ELEMENT Dorm (#PCDATA)> ]
```

Consider the following XQuery expression over documents conforming to the above DTD. We are omitting `doc(...)` specifications.

```xml
for $d1 in /University/Dept
for $d2 in /University/Dept[@Name <> $d1/@Name]
for $f in $d1/Faculty
let $s1 := $d1/Student[@Advisor=$f/@Name]
let $s2 := $d2/Student[@Advisor=$f/@Name]
where count($s2) > count($s1)
return <Mystery>
  <F>{$f/@Name}</F>
  <D>{$d2/@Name}</D>
</Mystery>
```
a. (8 points) State in English what is computed by this query. Your answer will be graded on clarity and conciseness, as well as on correctness.

b. (4 points) Is it possible to rearrange the for and let clauses in the above query so at least one let appears before at least one for, without changing the meaning or correctness of the query? If yes, provide such a query as evidence. If no, explain why.
c. (4 points) Can the above query produce duplicate **Mystery** elements in its result? Explain why.

d. (4 points) Describe what could change in the query result if the condition [@Name <> $d1/@Name] is removed from the second for clause. Your answer will be graded on clarity and conciseness, as well as on correctness.
Problem 2 (24 points)
Consider querying XML documents containing information about students in classes. The documents conform to the following DTD:

```xml
<!DOCTYPE Classes [ 
  <!ELEMENT Classes (Class*)> 
  <!ELEMENT Class (Topic, Students)> 
  <!ATTLIST Class Number ID #REQUIRED Units CDATA #REQUIRED> 
  <!ELEMENT Topic (#PCDATA)> 
  <!ELEMENT Students (Student+)> 
  <!ELEMENT Student (FirstNm, LastNm)> 
  <!ELEMENT FirstNm (#PCDATA)> 
  <!ELEMENT LastNm (#PCDATA)> ]>
```

For each of the query pairs in (a)–(e), decide if the XPath and XQuery expressions are equivalent (i.e., they are guaranteed return the same result over any XML document conforming to the above DTD). If you think they are, just write YES. If they are not equivalent (i.e., there is some document conforming to the DTD for which they will return different results), then write NO, give an example document to support your decision, and explain why the results for the XPath and the XQuery expressions are different. For equivalence don’t take into account details of answer presentation (such as `<result>` tags), just consider whether the query results contain the same set of elements. Also don’t worry about `doc(...)` specifications or type coercions.

a. (4 points)
XPath:  
//*[@Number="1234"]/Student

XQuery:
```xquery
for $c in /Classes/Class
where $c/@Number="1234"
return $c/Students/Student
```
Are the queries equivalent?
b. (4 points)
XPath:
/Classes/Class[Students/Student[LastNm="Smith"]]/Topic

XQuery:
for $c in /Classes/Class
where every $n in $c/Students/Student/LastNm satisfies $n="Smith"
return $c/Topic
Are the queries equivalent?

c. (4 points)
XPath:
/Classes/Class
[Students/Student/FirstNm != Students/Student/FirstNm
and Students/Student/LastNm != Students/Student/LastNm]/Topic

XQuery:
for $c in /Classes/Class
for $s1 in $c/Students/Student
for $s2 in $c/Students/Student
where $s1/FirstNm != $s2/FirstNm and $s1/LastNm != $s2/LastNm
return $c/Topic
Are the queries equivalent?
d. (4 points)

XPath:

/Classes/Class[@Units="5"][3]/Topic

XQuery:

for $c in /Classes/Class[3][@Units="5"]
return $c/Topic

Are the queries equivalent?
e. (4 points)
XPath:
/Classes//Student[1]

XQuery:
let $s := /Classes/descendant::Student[1]
return $s
Are the queries equivalent?

f. (4 points) For the same DTD, consider the following XQuery expression evaluated over a data set representing one class with 3 students. List all of the tag names that would appear in the result, including any duplicates.

    for $s in // (Students | Student)
    for $x in $s/preceding-sibling::*
    return name($x)

Tag names:
Problem 3 (56 points)

The CSE department has two databases containing information about students and courses. The first database contains information about undergraduate students and the courses they have taken. The root element is ugrads, with one or more student and enrollment subelements. Each student has a unique studentID and a name. Each enrollment has a courseID, a studentID and optionally the grade the student received. The second database keeps information about graduate courses and the students enrolled in them. The root element is gradCourses, with one or more course subelements. Each course has a unique courseID, a title and zero or more student elements. Each student has a studentID (unique within the enclosing course element), a name and a grade for the enclosing course. The CSE department hired you to integrate the two databases into a virtual students database. They told you that the root element has to be students with one or more student subelements. Each student should have a unique studentID, a name, a type, which should be either GRAD, for graduate students, or UGRAD, for undergraduate students, and zero or more course elements. Each course must have a courseID (unique within the enclosing student element), optionally a title, a type, which should be either GRAD, for graduate courses, or UGRAD for undergraduate courses, and a grade received by the enclosing student. The department told you that there are no data inconsistencies between the two databases, but information present in one database might be missing from the other.

a. (18 points) Define the XML Schemas for all three databases including the appropriate keys and key references. For the undergraduate students database, place your schema in ugrads.xsd. For the graduate courses database, place your schema in gradCourses.xsd. Place the integrated schema in students.xsd.

b. (38 points) Define in XQuery the GAV mapping from the two local databases to the virtual integrated one. Assume the undergraduate students database is stored in ugrads.xml and the graduate courses database in gradCourses.xml. Your answer will be graded on completeness, as well as on correctness. Place the GAV mapping in mapping.xq.