Consider the following DTD.

<!DOCTYPE University [

<!ELEMENT University (Student\*, Teacher\*,Course\*)>

<!ELEMENT Student (Name,Address+)>

<!ATTLIST Student

studentID ID #REQUIRED   
 courses IDREFS #IMPLIED

>

<!ELEMENT Name (#PCDATA)>

<!ELEMENT Address (Street,City)>

<!ELEMENT Street (#PCDATA)>  
 <!ELEMENT City (#PCDATA)>

<!ELEMENT Teacher (Name,Address+)>

<!ATTLIST Teacher

employeeID ID #REQUIRED  
 courses IDREFS #IMPLIED

>

<!ELEMENT Course (Name)>

<!ATTLIST Course

course ID #REQUIRED   
 teacher IDREF #REQUIRED

students IDREFS #REQUIRED

>

]>

It means that every university has some students, some teachers, and some courses. For every student, we store their name and address. Each student also has the attributes studentID and courses. The first attribute stores the ID of the student, while the second attribute contains the IDs of the courses that the student takes, separated by space. Note that all IDs inside an XML document must be different. That is, we cannot have a student that shares an ID with a course.

The name of the student is a string and the address of the student consists of Street and City, where both are leaf nodes. Every teacher has a name and an address. The attributes of a teacher node are the ID of the teacher (employeeID) and the courses they teach (courses). Finally, a course only contains the attributes course, teacher, and students.

1. [4] Draw an example XML tree for the DTD. Include several courses, teachers, and students. Remember that nodes cannot share IDs.
2. [4] Show the XML document for the tree.
3. [4] Write an XPath query that returns the IDs of the students that are taking the course with ID=”CSC366”. Use the contains(s1,s2) method to check if the string s2 appears in the string s1.
4. [4] Create a JSON file for the same example as Q2. Note that there is no distinction between the value of an attribute and the label of a node that can be reached by following an edge in JSON.
5. [4] Complete Q3 again, but this time query the JSON file.
6. Suppose that we have only three words in the language: a, b, and c, the documents:

d1: a a b a c a  
d2: b b a a c c a  
d3: a c c

and the query

q: a b.

a[2]) Show the text vectors for the query and the documents.

b[2]) Normalize the vectors using the TF-IDF formulas. Make sure to use the 0.5 formula for the query.

c[2]) Compute the cosine distance between the query and each of the documents using the normalized values.