CSE 462 Homework #5: Semi-Structured Data

Name: ____________________ Date: April 22, 2011

***** Due on Due on May 02, 2011 at the beginning of class. *****

This problem set is worth 250 points. You **must** type in your answers. Please, format your XML, DTD, XSchema, and XQuery.

Consider the reference *relational schema* for an online store given below.

Customers(cemail: *string*, name: *string*, phone: *string*, street: *string*, city: *string*, state: *string*, zip: *string*)

Products(sku: *int*, name: *string*, unitPrice: *real*, unitWeight: *real*)

Orders(oId: *int*, cemail: *string*, orderDate: *date*, shipDate: *date*)

OrderEntries(oId: *int*, sku: *int*, qty: *int*)

Shipments(oId: *int*, street: *string*, city: *string*, state: *string*, zip: *string*)

Observations:

- Keys are underlined.
- `phone` in `Customers` may be `NULL`.
- `unitWeight` in `Products` may be `NULL`.
- `shipDate` in `Orders` may be `NULL`.
- `cemail` in `Orders` references `cemail` in `Customers`.
- `oId` in `OrderEntries` references `oId` in `Orders`.
- `sku` in `OrderEntries` references `sku` in `Products`.
- `oId` in `Shipments` references `oId` in `Orders`.
- The *entry count* of an order is the number of entries associated with the order.
- The *item count* of an order is the sum of product quantities over all entries in the order.
- The *gross value* of an order is the sum of `qty` × `unitPrice` over all entries in the order.
- The *gross weight* of an order is the sum of `qty` × `unitWeight` over all entries in the order.

The online store is in the process of developing an application to provide information to the main office as XML. Your job is to show alternate approaches in which this can be accomplished. For this, you will experiment with both DTD and XSchema for representing the data. You will also show some use cases of how the XML data can be queried using XPath and/or XQuery.

XSchema Type Reference:

- `http://www.w3schools.com/schema/schema_dtypes_string.asp`
- `http://www.w3schools.com/schema/schema_dtypes_date.asp`
- `http://www.w3schools.com/schema/schema_dtypes_numeric.asp`
- `http://www.w3schools.com/schema/schema_dtypes_misc.asp`

DTD, XSchema, and XML Validation:

- `http://www.xmlvalidation.com/`

1. (75pts) Provide a DTD (30pts) and an XSchema (30pts) to encode the reference relational schema based on the requirements described below. Then, provide a minimal XML document (15pts) that is valid with respect to the DTD but not with respect to the XSchema, if one exists.

- The root of your XML document is a `store` element containing `customers`, `products`, and `orders` elements.
- The `customers` element contains any number of `customer` elements. Every `customer` element represents a tuple in the `Customers` relation.
- The `products` element contains any number of `product` elements. Every `product` element represents a tuple in the `Products` relation.
- The `orders` element contains any number of `order` elements. Every `order` element represents a tuple in the `Orders` relation. Each `order` element must contain at least one `orderEntry` element. Every `orderEntry` element represents a tuple in the `OrderEntries` relation associated with the respective `order`. Each `order` element may contain one `shipment` element. Every `shipment` element represents a tuple in the `Shipments` relation associated with the respective `order`.
- Do not include `oId` values for `order`, `orderEntry`, and `shipment` elements. The relationship among these elements are determined from the document structure.
- For all other keys/foreign keys in the relational schema, try to reproduce them in the DTD and XSchema as best as possible, including their data types.

2. (75pts) Provide a DTD (30pts) and an XSchema (30pts) to encode the reference relational schema based on the requirements described below. Then, provide a minimal XML document (15pts) that is valid with respect to the DTD but not with respect to the XSchema, if one exists.

- The root of your XML document is a `store` element containing `products` and `customers` elements.
- The `products` element contains any number of `product` elements. Every `product` element represents a tuple in the `Products` relation.
- The `customers` element contains any number of `customer` elements. Every `customer` element represents a tuple in the `Customers` relation. Each `customer` element may contain any number of `order` elements. Every `order` element represents a tuple in the `Orders` relation associated with the respective `customer`. Each `order` element must contain at least one `orderEntry` element. Every `orderEntry` element represents a tuple in the `OrderEntries` relation associated with the respective `order`. Each `order` element may contain one `shipment` element. Every `shipment` element represents a tuple in the `Shipments` relation associated with the respective `order`.
- Do not include `oId` or `cemail` values for `order` elements; do not include `oId` values for `orderEntry` and `shipment` elements. The relationship among these elements are determined from the document structure.
- For all other keys/foreign keys in the relational schema, try to reproduce them in the DTD and XSchema as best as possible, including their data types.

3. (50pts) Write an XQuery expression that takes as input an XML document valid with respect to the DTD of problem 1 (call it 'input.xml'), and outputs an XML document valid with respect to the DTD of problem 2. All information contained in the input document must be represented in the output document.

4. (50pts) Write XQuery expression to answer each of the problems below. These problems refer to the 'phd.xml' document, which can be found on the course web page with their respective DTDs.

- (25pts) Find all PhD recipients having the largest number (say, *N*) of proper ancestors. Return a `deepest` element containing an attribute `depth` with the value of *N*. The `deepest` element should also contain one child `recipient` element for each recipient having *N* proper ancestors. For each such recipient, return the recipient element with its `id` attribute and its child `name` element; *do not return* the recipient's `phd` or `recipients` child elements.
- (25pts) For every country in which some recipient received a PhD, return a sequence of `country` elements ordered alphabetically by country name. For each such `country` element, include a `name` element with the country's name and one `classOf` element for every year in which the country had a PhD. Each `classOf` element must contain two attributes: the `year` of the "class" and the `total` number of PhD recipients for that year and country. Within the `classOf` element, return the `name` elements of the respective PhD recipients.