CSE 460 Project 3: TinyKayak: XML and XQuery

Summer 2019

Instructor:	Zhengxiong Li	**Time:**	MW 9-12PM, 7.7-8.14
Email:	zhengxio@buffalo.edu	**Place:**	Baldy 117.

Submission:
Failure to comply with the submission specifications will incur penalties for **EACH** violation.

- What to submit: A zip file has to be submitted to the Email '*zhengxio@buffalo.edu*' by **8/9/2019 11:59PM EST** with the email title *CSE460_proj3_ubit* (for example: *CSE460_proj3_jack*, where *jack* is the ubit). Only **zip and tar** ex-tension will be accepted, please don't use any other compression methods such as 7zip.
- Zip file naming: Use *proj3_ubit* (NO SPACE!) for the filename, for example: *proj3_jack*.zip, where *jack* is the ubit. The project is an **INDIVIDUAL** project, the filename should contain only one ubit.
- Sub-structure of zip file: On unzipping the zip file, there should be a folder named with your ubit *proj3_ubit*, under the folder *proj3_ubit*, there should be one file, a **.txt file** containing your answers for all the problems.

Problem 1: XML Database Design
You are asked to design an XML database for TinyKayak, which is a travel booking website. TinyKayak has three types of data to manage: customer data, airline information and flight ticket booking data. Specifically, you are given the following list of requirements:

- TinyKayak creates a record for each customer, and each record consists of:
  1. A passport number, the passport number is also the key for the customer records.
  2. A legal name.
  3. A date of birth (DOB).
  4. At least one address, each address consists of street, city, state, and zipcode.
  5. A list of flight ticket orders that belong to the customer. (The requirements of order data will be given soon).
- TinyKayak creates a record for each airline, and each record consists of:
  1. An airline code, which is also a key.
  2. An airline name.
- TinyKayak also creates a record for each flight ticket order, and each record consists of:
  1. An order number, the order number is also the key for the flight ticket order records.
  2. At least one airline code, airline codes indicate which airline companies the order associates to.
  3. A payment type, which indicates if the order is paid by a credit card, a check, or a debit card, note only these three types are valid types for payment type.

Write a DTD to model the above requirements. Use any xml validation tool to validate your DTD with your test data, e.g., an XML Validation tool can be seen at https://www.xmlvalidation.com/. For this problem, your solution should include your DTD and your test data that is valid against your DTD (note: test data shouldnt be empty, and your test data should be able to show all the modeling components of your DTD).

Problem 2: XQuery

Given the following DTD describes the information of the buildings of a university: *department* indicates the departments that are using the buildings, *name* is the name of the building, *type* indicates how the building is used, e.g., teaching means that the building is used for teaching purpose, *year* is the year the building was built. Assume you have a xml database called *buildings.xml* that is valid against the given DTD, write the following queries in your solution file, use XQuery comments to separate your answers, e.g., (: answer for 2.1 :). Use eXistDB to test and verify your answers, the file path of buildings.xml should be **/db/buildings.xml**. *Note: Your solution should include only your queries, please do not include the given DTD or any test data.*

```
<! DOCTYPE buildings [
  <! ELEMENT buildings (building*) >
  <! ELEMENT building (department+, name, type+, year) >
  <! ATTLIST building id ID #REQUIRED >
  <! ELEMENT department (#PCDATA) >
  <! ELEMENT name (#PCDATA) >
  <! ELEMENT type (#PCDATA) >
  <! ELEMENT year (#PCDATA) >
] >
```

- Find all the buildings which are used only as a library, i.e., the type is "library".
- Find all the buildings which are used by both CSE department and EE department.
- For every department, find the department name and the total number of buildings built in 2000 the department is using, you can ignore those departments that are not using any building built in 2000, the result should be sorted in lexicographical order by the department name.