ECS713 group project

JSON to XML translation using Aeson

Due Date: Tuesday 11th November 2014

Electronic submission through college system

Summary: The project consists of writing a JSON to XML translator using the Haskell module **Aeson**. Your program should work on one of the many JSON datasets available online. The core component of your program should be the function from the JSON representation to the XML format, via an intermediate Haskell datatype.

Aeson package:
https://hackage.haskell.org/package/aeson-0.8.0.0/docs/Data-Aeson.html

Background on the JSON and XML format: You should all be familiar with JSON and XML formats, but if not, then please read the w3 tutorials at

http://www.w3schools.com/xml
http://www.w3schools.com/json

and in particular the JSON sections Home, Intro and Syntax (the remainder is largely about using JSON in connection with Javascript and browsers), and the XML sections on syntax, elements and attributes. The formal specification of XML 1.0 documents is available from the w3c at

http://www.w3.org/TR/2008/REC-xml-20081126

You will not need to read all of this, but the document gives the syntax in Extended Backus-Naur Form (EBNF), which allows a relatively simple translation from Haskell datatypes, see section 6: http://www.w3.org/TR/2008/REC-xml-20081126/#sec-notation
This is the core specification that will allow you to ensure that you produce properly compliant xml.

Part 1: Parse JSON input into Haskell data type (using Aeson)

Once you have chosen a particular JSON dataset, you should

- document the structure of the JSON dataset, concentrating on how the set is structured as a tree, and what types of information is included (this documentation should be included in your submission)
- devise a Haskell data type that corresponds to the structure of the JSON file you are using

- write a function (using Aeson) to navigate the JSON input and extract the strings giving the basic data
- convert JSON input string into an element of the corresponding Haskell data type

See the following book section on creating new Haskell data types:

http://book.realworldhaskell.org/read/defining-types-streamlining-functions.html

Part 2: Convert Haskell data type into XML format

Once the data is availabe as a Haskell data type, you should then

- write a function that converts the Haskell data type into XML format
- combine this with the function from Part 1 so as to provide a converter from JSON to XML

Part 3: Write at least three functions to query parsed data

Provide at least three function that allows the user to query the input data. For instance, if reading data about stock prices, the program could calculate highest and lowest prices, (2) average prices, etc. Your program and the different functions will be tested on ghci, so there is no need for a "main" function that interacts with the user.

Assessment criteria: The following are going to be the main assessment criteria used to grade your project:

- ***Functionality***, code compiles and performes tasks correctly. (40%)
- ***Knowledge of Haskell***, which includes proper uses of **data** and **type**, definition by cases, recursion, etc. (20%)
- ***Structure and quality of code***, this includes type declarations of all functions, and clear and concise comments throughout the code. (15%)
- ***Additional features,*** any extra features that go beyond the minimum required above. (25%)

Assistance: See also Real World Haskell (Chapter 5)

http://book.realworldhaskell.org/read/writing-a-library-working-with-json-data.html

for an account of JSON datatypes and rendering.