CSCI 3020 ASSIGNMENT 2 (50 POINTS)

DUE THURSDAY, SEPTEMBER 17, 11:59 PM

GOALS

This assignment has you convert another CSV file into XML. You will use XML namespaces in the XML output.

GETTING HELP

- Your textbook from CSCI 1010/2010 covers most of the C++ that is necessary for this assignment. We will cover additional C++ in lecture.
- Post general questions on the discussion board in D2L. Please refrain from posting more than 1 line of code, otherwise your post may be removed.
- Send me email, nicholsonja@apsu.edu, or come to my office. I'll be happy to help. If emailing me, please do not send me a code snippet; send your entire .cpp so that I can compile and test it.

BACKGROUND

The Small Town Library is updating their files. They have asked you for help in converting their data. One of the files is a reference file the librarians use when they are asked questions about population of cities around the world.

The librarians have provided you with two files **data-1.csv** and **data-2.csv** that contain example records from the library in the old format. The data-1.csv file is a short file that contains just a few records. It is short sample of the type of data you the file data-2.csv is a longer file. It may contain some types of data that data-1.csv does not.

DATA FILE FORMAT

The library's data files are stored in a format called CSV, which stands for **comma separated values**. It is another yet another common text file format. It is an oldie, but goodie, and is still used today. One downside though is that it provides limited semantic information.

Although CSV is yet another file format, the basic rules for CSV and XML are both the same:

- If you can read and/or write a text file, you can read and/or write a CSV file. CSV files are just text files.
- If you can read and/or write a text file, you can read and/or write an XML file. XML files are just text files.
- Specific file formats, even text-based formats, have specific rules. For example, XML has tags, attributes, and elements, and CSV has commas. But the contents are still text.
- File extensions – .txt, .csv, .xml – are only mildly important. If you can open a file in NotePad++, Sublime, vi, or any other text editor, they are text files, regardless of what the 3-letter extension is.

In the CSV files you have been given, each line is a record representing information about a single city. Each line contains 6 fields, and a comma separates the fields. The format of each line, or record, is:

Country-Name,Country-Region,Country-Code,City-Name,City-District,City-Population

For example, the first line of a data file might look like this:

Denmark,Nordic Countries,DNK,Aalborg,Nordjylland,161161

That means the info on that line can be broken up into the following pieces of data:

- Country name: Denmark
- Country region: Nordic Countries
- Country code: DNK
- City name: Aalborg
- City district: Nordjylland
- City population: 161161

It is important to note that the commas are not part of the data; they are simply used to separate the fields. Every line in both CSV files follows this format.

MAIN REQUIREMENTS

Your program needs to convert the old CSV format to a new XML-based format. Basically, your program should:

- Prompt the user for a text-based data file to read.
    - If there is a problem opening the file, inform the user with an appropriate error message and then exit.
    - Your program should be able to accept any input file, not just files named data-1.csv and data-2.csv.
    - You can assume the input file given is a CSV file in the correct format.
- If you can successfully open the input file, prompt the user for the XML that should be written to.
    - If the filename does not end in ".xml" you should add it to the filename.
    - If user's filename already ends in ".xml", you should not add anything to the filename.

When the filenames are considered good, the program should convert the CSV data to the following XML format:

- Start with the XML prolog

  ```
  <?xml version="1.0" encoding="UTF-8"?>
  ```

- The next line should be a comment with your name in the following format

  ```
  <!-- Processed by John Nicholson's converter -->
  ```

  You should obviously replace my name with your name.

- The remaining lines should be tags and data. There should not be any attributes in any element.
    - The root elements should be **countries**
    - Inside the **countries** element, should be series of **country** elements.
        - A country has a **name**, a **code,** and a **region**. All three should be elements.
        - A country also has a **cities** tag. This tag contains a list of ALL the cities in the country.
            - For each city in **cities**, there will be a **city.** A **city** has a **name**, a **district**, and a **population.**

          Please note the requirement that a **country** element contains all the cities listed for that country.

          For example, in data-1.csv, Denmark has 3 cities, and the Dominican Republic has 2 cities. In the XML output, there is 1 country tag for Denmark, and inside its **cities** tag are 3 **city** tags. There is 1 **country** tag for the Dominican Republic, and inside its **cities** tag are 2 **city** tags.

          Denmark should not appear under 3 country tags, and the Dominican Republic should not appear under 2 country tags

The library provided a sample XML containing the format they want. The sample XML file is part of your download, **sample_output.xml**. It is how the data in data-1.csv should look after your program processes it. The output for your program should

- Produce the same elements in the same order.
- Make sure the value in the **population** has a comma. The librarians really want commas in their numbers for readability purposes. For example, output this (see the comma in the number):

  <population>161,161</population>

  Do not output this (no comma in the number):

  <population>161161</population>
- Indentation is optional. You can completely leave it out if you wish, or put it in. Either is fine.

Unfortunately**,** there is a problem with sample XML that needs to be addressed.

The library staff does not have experts in XML. Two of the elements they want – the **name** element for the country and the **name** element for the city – are the same. That is a problem because a city name and a country name are not semantically the same thing. Therefore, in your position as a technical consultant, you have requested that the provided XML format should be modified. The modifications[1] are:

- All the tags related to country information - **countries**, **country**, **name**, **code,** and **region** - should all be placed in an XML namespace named **country**
  - Use the URL **http://library.smalltown.us/country**
- All the tags related to city information – **cities, city, name, district,** and **population** – should all be placed in an XML namespace named **city**
  - Use the URL **http://library.smalltown.us/city**
- No tag in the document should be in the default namespace.

The librarians are totally cool with this change. This means the output of your program, when finished, will be structured the same as the sample_output.xml, but it will look slightly different. You must use XML Namespaces to resolve this element conflict with the **name** elements.

The XML document generated by your program must be well-formed. After you generate an XML output file, open it in jEdit and test it using the menu option **Plugins -> XML -> Parse as XML**. The file you generate should produce no errors.

[1] The URLs given in the modifications are not real websites. Don't worry that your browser will not be able to find them. That does not affect the XML you will be generating.

TIPS

- The CSV file extension is typically associated with spreadsheet programs like Excel. You will need to examine the file in a text editor, so make sure you do an "Open With…" selection if you are opening the file from your desktop.
- Watch out for characters that need to be encoded.
- Listen in lecture and watch for a video. I will demonstrate a way in C++ to deal with the commas when reading the CSV file.

ADDITIONAL C++ REQUIREMENTS

Additional requirements

1. You should not have one, ginormous main() function. You program should be modular, that is, define several (more than 2) meaningful functions. All functions should have a comment describing their purpose.
2. You may not use any global variables.
3. You may not use cstring, that is, do not do this:
   #include <cstring>
   You can use string, that is, this is ok:
   #include <string>
4. You may not declare any variable or parameter as a character pointer, **char ***, but you can of course use the **string** type.
5. You may not use the **exit()** function.
6. Except for items #3, #4, and #5 above, you may use any feature of C++ defined in the language or STL[2]. Your CSCI 1010/2010 book covers the basic necessary C++ needed for this assignment.
7. You may not use any non-standard C++ library or feature specific to your compiler. For example, Visual Studio users may not use the XML libraries provided by Microsoft.
8. You must include a comment at the top of your main.cpp file in this format:

```
/*******************
John Nicholson
CSCI 3020 Section 09
Fall 2015
Assignment 1

Programmed on Windows 7 using Visual C++ 2010 Express

This program converts ...
*******************/
```

   If this comment is missing, you will receive 0 points on the assignment. You should obviously change the name to your name, the section to your section, the system to the system you used, and write a real description of the program. You must include the specific operating system and the compiler/IDE you used.

[2] Feel free to ask me why I have the requirements #3, #4, and #5. It isn't just to make your life difficult, really.

Please note that **Windows** is not an operating system; **Windows 7**, **Windows 8**, and **Windows 8.1** are operating systems. **Visual Studio** is not a compiler/IDE. **Visual C++ 2010 Express** and **Visual Studio Ultimate 2013** are compiler/IDEs. You should be specific about which OS and compiler/IDE you used, otherwise you may lose points in grading because I graded using the wrong compiler.

NOTE: I will not regrade if you do not put in the correct comment or are not specific enough, and I end up testing your code with a different compiler than you did causing your code to crash.

SUBMITTING

Upload the following three files to the Assignment 1 DropBox in D2L:

- Your .cpp file
- The XML file generated by your program from processing data-1.csv
- The XML file generated by your program from processing data-2.csv

Do not upload your entire Visual Studio project (or project of whatever you are using). Just upload the three files listed above.