CIT 483/583 Lab06 – REXML

**Instructions**

Open and save this file in any MS word-compatible format as Lab06\_*Firstname*\_*Lastname*.<ext> and place your answers in that document. Do a **Save-As** and retain all of my content. Keep the document safe in case your submission fails, or you discover an error prior to the due date and wish to re-submit. Submit your document to the Lab06 dropbox in Canvas. The due date and any other pertinent information are noted in the Canvas item.

This lab should be completed on students.cs.nku.edu as a reference implementation and as a means of testing your answers. It can be done using IRB only, or you may create .rb files to test some or all of your code, but do not submit .rb files (programs).

**Place the answers in or immediately following each question and make sure your answers stand out from the questions by using a different font color.**

1. The following XML is not well-formed. Identify three (simple) changes that can be made to correct it. [The choice of single vs. double quotes is not an issue as long as they match in pairs.]

<receive>

<car make="Ford" model='Focus' year='2012">

<display>2012 Ford Focus</display>

</car>

<car make="Chevy" model='Impala' year="2009">

<display>2009 Chevy Impala<display>

</car>

</recieve>

a)

b)

c)

2. Copy ~mccordt/master\_scoreboard.xml on students.cs.nku.edu to your CIT483 directory. Use irb to perform the following interactions and capture all output.

$ irb --simple-prompt

>> require 'rexml/document'

>> include REXML

>> dom = Document.new(File.open 'master\_scoreboard.xml')

>> dom.class

>> dom.root.class

>> dom.root.elements.class

>> dom.root.elements.size

>> dom.root.children.size

>> dom.root

>> dom.children

>> dom.children.class

>> dom.children[0]

>> dom.children[1]

>> dom.children[3]

>> dom.children[3] == dom.root

>> dom.elements[1] == dom.root

>> dom.elements.to\_a('/games/game').size

>> dom.elements.to\_a('/games/game/winning\_pitcher/').size

>> dom.elements.to\_a('/games/game/winning\_pitcher/')[0]

>> dom.elements.to\_a('/games/game')[3,2].each do |gg|

gg.elements.each {|el| p el.name }

end

[Abbreviate the output to 10-12 lines]

>> dom.elements.to\_a('/games/game/winning\_pitcher/').each do |p|

puts "Last name: #{p.attribute('last')} ERA: #{p.attribute('era')}"

end

[Abbreviate the output to the first 10-12 lines]

>> dom.elements.to\_a('//losing\_pitcher[@last="Villarreal"]')

>> query = '//losing\_pitcher[@last="Villarreal"]/../winning\_pitcher'

>> dom.elements.to\_a(query)[0].attribute("last")

3. Use Ruby and ReXML to answer the following questions. Show the Ruby code used to find the answer as well as the answer. You may abbreviate the result if it is more than 10-12 lines.

NOTE: You may continue your irb session from the previous question for testing; otherwise, you will have to load the required extensions and objects again, and initialize the DOM object again:

>> require 'rexml/document'

>> include REXML

>> dom = Document.new(File.open 'master\_scoreboard.xml')

The elements method allows you to reference items by name (not just by number). For example, to get the first game's winning\_pitcher attributes:

>> dom.elements.to\_a('//game')[0].elements['winning\_pitcher']

=> <winning\_pitcher id='476454' last='Betances' first='Dellin' name\_display\_roster='Betances' number='68' era='1.47' wins='6' losses='2'/>

To get selected winning \_pitcher elements, use [@attr = value] where = can be any relational operator:

dom.elements.to\_a('//winning\_pitcher[@wins >10 ]')

a) Who are the winning pitchers whose ERA is greater than 2?

b) Who are the winning pitchers whose ERA is less than 2?

c) Who are the winning pitchers who have fewer than 10 wins?

d) Who are the losing pitchers who have more than 5 wins??

e) What is the total sum of wins for all winning pitchers?

HINT: Use the starter code below that iterates over all winning\_pitcher elements and extracts the number of wins, which is a a REXML::Attribute. To treat it as a number, first get its String value, then call to\_i. Create a counter and, each time you get a number, add it to the counter.

total = 0

dom.elements.to\_a('/games/game/winning\_pitcher/').each do |p|

# process wins as described and add it to total

puts p.attribute('wins')

end

puts total

4. The following script contains a "listener" class that is based on the version in the lecture slides. It is written to isolate information on winning\_pitchers . Enter the code directly into IRB and answer the three questions that follow. You do not need to show IRB output.

require 'rexml/document'

require 'rexml/streamlistener'

include REXML

class MyListener

include StreamListener

def tag\_start (name, attr)

if name == "winning\_pitcher"

puts "Winning pitcher: #{name.inspect} −− #{attr.inspect}"

end

end

def tag\_end(name)

# puts "tag end: #{name.inspect}"

end

def text( val)

# puts "text : #{val.inspect}"

end

def comment(cmt)

# puts "comment: #{cmt.inspect}"

end

def entity (ent)

# puts "entity : #{ent.inspect}"

end

end # class

Load the class in IRB and then run:

file = File.open 'master\_scoreboard.xml'

Document.parse\_stream(file, MyListener.new)

NOTE: The way this code is written, the only way to find specific data or count things is to inspect the output manually. It is meant to demonstrate the difference between DOM and SAX parsing.

Also, if you need to reread the document, you have to re-open the file each time, as in:

file = File.open 'master\_scoreboard.xml'

Document.parse\_stream(file, MyListener.new)

a) Who are the winning pitchers whose names begin with G"?

b) Who are the winning pitchers whose names begin with A, B, C, or D"?

c) Who has the highest number of losses among the winning pitchers?