CS 2316

Exam 4 Spring 2012

Name : ____________________

Grading TA: ____________________

- Integrity: By taking this exam, you pledge that this is your work and you have neither given nor received inappropriate help during the taking of this exam in compliance with the Academic Honor Code of Georgia Tech. Do NOT sign nor take this exam if you do not agree with the honor code.
- Devices: If your cell phone, pager, PDA, beeper, iPod, or similar item goes off during the exam, you will lose 10 points on this exam. Turn all such devices off and put them away now. You cannot have them on your desk.
- Academic Misconduct: Academic misconduct will not be tolerated. You are to uphold the honor and integrity bestowed upon you by the Georgia Institute of Technology.
  - Keep your eyes on your own paper.
  - Do your best to prevent anyone else from seeing your work.
  - Do NOT communicate with anyone other than a proctor for ANY reason in ANY language in ANY manner.
  - Do NOT share ANYTHING during the exam. (This includes no sharing of pencils, paper, erasers).
  - Follow directions given by the proctor(s).
  - Stop all writing when told to stop. Failure to stop writing on this exam when told to do so is academic misconduct.
  - Do not use notes, books, calculators, etc during the exam.
- Time: Don't get bogged down by any one question. If you get stuck, move on to the next problem and come back once you have completed all of the other problems. This exam has 6 questions on 10 pages including the title page. Please check to make sure all pages are included. You will have 50 minutes to complete this exam.

I commit to uphold the ideals of honor and integrity by refusing to betray the trust bestowed upon me as a member of the Georgia Tech community. I have also read and understand the requirements outlined above.

Signature: ____________________

Question	Points	Score
1. Vocabulary	9	
2. Creating a Child	6	
3. Multiple Choice	4	
4. SQL Sales	24	
5. Ant XML	8	
6. Soccer XML	18	
Total:	69	

1. *(9 points)*
For each of the following vocabulary terms, write a concise 1-2 sentence definition. Be brief, and to the point.

   (a) [3 pts] flow of execution

   Solution: The order in which statements in a program are executed. Function calls, return statements, conditionals and loops all modify the standard top to bottom flow of execution.

   (b) [3 pts] parameter

   Solution: parameter - A name used inside a function to refer to the value passed as an argument.

   (c) [3 pts] element (in XML)

   Solution: In XML, an element is a collection of data that is delimited by start and end tags. Elements may contain attributes within the start tag, text between the start and end tags, as well as subelements. Looking for: (start/end tags or named by tags, attributes, text, contain subelements)

2. *(6 points)*
Given the code below, write a python code example showing two DIFFERENT ways to create and attach a subelement to the root element. Use tag names of 'child1' and 'child2' for the two elements. Set the TEXT of child1 to be "Kid 1" and add an attribute of kidNumber="2" for child2.

```
import xml.etree.ElementTree as etree
root = etree.Element("MyRoot")
```

Solution:

```
x = etree.SubElement(root, "child1")
x.text = "Kid 1"

y = etree.Element("child2")
y.attrib['kidNumber'] = '2'  # or y = etree.Element("child2", kidNumber="2")
root.append(y)

Grading:
+2 for creating a SubElement directly.
+2 for creating an Element, and then appending it to the root.
  (It does not matter which method they use for which child!)
+1 for setting child1's text to be "Kid 1"
+1 for adding an attribute of kidNumber="2" to child2
```

3. *(4 points)*
For each of the following multiple choice questions, indicate the most correct answer! Indicate your selected answer by circling it.

(a) [1 pt] Which of the following ways will correctly place attributes in an XML Element? You may assume etree is the correct import.

A.
```
myDict = {"key1": "value1"}
element = etree.Element("tag", myDict)
```
B.
```
element = etree.Element("tag", key1="value1", key2="value2")
```
C.
```
element = etree.Element("tag")
element.get("key1", "value1")
```
D.
```
element = etree.Element("tag")
element.set(value1 = "key1")
```

(b) [1 pt] Which of the following statements is correct of proper XML?

A. A particular XML tree may posess only one root.
B. An element may have only one child.
C. Every element must have an opening and closing tag.
D. An element may not have an attribute named 'text'.

(c) [1 pt] Given the following line of XML, what is the correct term for "mystery1"?
```
<mystery1 mystery2="mystery3">mystery4</mystery1>
```
A. Tag
B. Attribute
C. Text
D. Element

(d) [1 pt] Every element in an XML tree has at least one parent.
A. True **B. False**

4. *(24 points)*
A table has been created for you with the following command:

CREATE TABLE Sales (Id INTEGER NOT NULL AUTO_INCREMENT, Name TEXT, Email TEXT, Demand INTEGER, Order INTEGER)

(a) [4 pts] Write the SQL command that will return (only) the name of all people who have an email address ENDING in .edu:

Solution:

```
SELECT Name FROM Sales WHERE Email LIKE "%.edu";
```

Grading:
+1: SELECT Name FROM Sales
+1: WHERE Email
+2: LIKE '

(b) [3 pts] Write the SQL command that will return the calculated ratio between the Demand and Order integers (the ratio is calculated as Demand / Order), naming the returned column "Ratio".

Solution:

```
SELECT Demand / Order AS "Ratio" FROM Sales;
```

Grading:
+1: SELECT Demand / Order
+1: AS "Ratio"
+1: FROM Sales

(c) [5 pts] Write the SQL command that will return the name of each person in the table, along with the average demand for that person. (Each person may have multiple records, each with a different demand number.) Sort your results by the average demand in ascending order.

Solution:

```
SELECT Name, AVG(Demand) FROM Sales GROUP BY Name ORDER BY AVG(Demand)
```

Grading:
+1: SELECT Name
+1: AVG(Demand)
+1: FROM Sales
+1: GROUP BY Name
+1: ORDER BY AVG(Demand)

(d) [2 pts] Write the SQL command that delete all records from the table.

Solution:

```
DELETE FROM Sales
```

Grading:
+1: DELETE
+1: FROM Sales

(e) [10 pts] Write PYTHON CODE that connects to the "db.example.com" database host, with the username 'cs2316' and password 'SECRET', accessing the 'cs2316db' database. Then, execute the following SQL querry: "SELECT * FROM Sales". Take the results you receive back from the database server and PRINT them to the screen, but OMIT the first (Id) column from your printed output!

Solution:

```
import pymysql
db = pymysql.connect( host = 'db.example.com', passwd = 'SECRET',
                            user = 'cs2316', db='cs2316db')
c = db.cursor()
c.execute("SELECT * FROM Sales")
for item in c:
   print( item[1:] )
c.close()
db.close()
```

Grading:
+1: importing pymysql
+2: Correctly connecting to the DB (0.5 points per parameter)
+1: Getting a cursor
+1: Executing the querry.
+2: Using Fetchall or itterating through the cursor to get the results
+2: Printing all but the first element in each record (with slice or otherwise)
+1: Closing both the cursor and the db object when finished!

5. *(8 points)*
Here is a simple XML file:

```
<tick>
  <hillInfo>
    <x>90</x>
    <y>43</y>
    <ants>4</ants>
  </hillInfo>
  <antInfo>
    <ant x='91' y='44'>F W W W F FA F FH</ant>
    <ant x='90' y='45'>F FA F F W EA EA W</ant>
    <ant x='89' y='50'>W W EA EA EA F EA W</ant>
  </antInfo>
</tick>
```

Assume that the filename variable correctly indicates the above XML file. Write exactly what is printed when the code below is executed:

```
import xml.etree.ElementTree as etree
tree = etree.parse(filename)
root = tree.getroot()

for child in root:
   print(child.tag)

for child in root.find("antInfo"):
   print(child.attrib["x"])

for child in root.find("hillInfo"):
   print(child.text)

for child in root.findall("ant"):
   print(child.attrib["y"])
```

Solution:

```
Answer:
hillInfo
antInfo
91
90
89
```

```
90
43
4
```

Grading:
Each correct line in correct order +1
Each incorrect line mixed in with correct lines: -1 (not counting ant y values below)
Incorrectly having ant y values: (44,45,50): -4

6. *(18 points)*
Examine the following python code. Write out the textual XML file (soccer.xml) that is written to disk after it is executed. You must indent children under their parents to show the containment relationship (in addition to having the start and stop tags in the correct places.

```
import xml.etree.ElementTree as TR

soccerplayer =(   ( ("Frank Lampard", "11"), ("Didier Drogba", "5") ),
                  ( ("Sergio Aguero", "21"), ("David Silva", "6")   )    )

root = TR.Element("soccerteam")

t1=TR.SubElement(root,"team2",name="Manchester City",rank=str(2))
t2 = TR.Element("team1", name="Chelsea",rank=str(6))

count=0
for ss in soccerplayer[0]:
    print(ss)
    TR.SubElement(t1, "player", number=str(count), score=ss[1]).text=ss[0]
    count= count+1

for ss in range(len(soccerplayer[1])):
    nameplayer=soccerplayer[1][0]
    p2=TR.SubElement(t2,"player",number=str(ss))
    p2.text=nameplayer[ss]
    TR.SubElement(t2,"goals",score=soccerplayer[1][1][ss])

root.append(t2)
tree = TR.ElementTree(root)
tree.write("soccer.xml","UTF-8")
```

Solution:

```
<soccerteam>
  <team2 name="Manchester City" rank="2">
     <player number="0" score="11">Frank Lampard</player>
     <player number="1" score="5">Didier Drogba</player>
  </team2>

  <team1 name="Chelsea" rank="6">
     <player number="0">Sergio Aguero</player>
     <goals score="David Silva"/>
     <player number="1">21</player>
     <goals score="6"/>
  </team1>
</soccerteam>
```

Rubric:
+2 for each element that is fully correct (18 pts total)
(They must include text, attributes and ending tag locations) Specifically we are looking for: Soccerteam, team2, player, player, team1, player, goals player, goals.
For each element, take off -1 if an attribute and/or text is missing or incorrect but element is in right place.
Also take of -1 for any extra element that shouldn't exist!
Also take off -2 total if their indentation does not indicate that they understand the parent/child arrangement, even if the tags are ordered correctly.