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 5 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. Multiple Choice	4	
3. XML Writing	5	
4. Baseball	26	
5. XML Thrones	10	
Total:	54	

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] dictionary

   Solution: A mutable compound data type that associates keys with values. They map keys, which can be any immutable type, to values, which can be any type, just like the values of a list or tuple.

   (b) [3 pts] delimiter

   Solution: A character that is used to separate (or delimit) separate data items in a string. The default delimiter for comma separated value (CSV) files is the comma.
   Looking for: Separation of data items.

   (c) [3 pts] module

   Solution: A file containing Python definitions and statements intended for use in other Python programs. The contents of a module are made available to the other program by using the import statement.

2. *(4 points)*
   For each of the following multiple choice questions, indicate the single most correct answer! Indicate your selected answer by circling it.

   (a) [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

(b) [1 pt] Which of the following statements about SQL is true?

A. The syntax "DELETE *" is valid.

B. All keywords must be capitalized.

C. If specified, WHERE must come before FROM.

D. The % character is a wildcard when used with LIKE.

(c) [1 pt] Given the following code:

```
request = urllib.request.urlopen("http://www.google.com"):
```

A. `print( request )` will output the HTML

B. `print( html(request) )` will output the HTML

C. `print( request.read() )` will output the HTML

D. `print( str(request) )` will output the HTML

(d) [1 pt] Which of the following input does **not** match this regular expression:

`(\d\d+\d)*\..[a-z]{2}`

A. 12321.boo

B. ..ek

C. 9687.ra

D. 555.ahh

E. None of the above

3. *(5 points)*

Pretend you are the Python interpreter and the following code has been entered and executed:

```
import xml.etree.ElementTree as etree

x = etree.Element("People")
j = etree.SubElement(x,"Person",  age="77", classroom="C241")
j.text = "Jay Summet"
j = etree.SubElement(x,"Dog", age="5", breed="Pug")
j.text = "Snoopy"

t = etree.ElementTree(x)
t.write("test.xml")
```

Write out the (textual) contents of the test.xml file below:

Solution:

```
<People>
  <Person age="77" classroom="C241">Jay Summet</Person>
  <Dog age="5" breed="Pug">Snoopy</Dog>
</People>
```

Grading:
+1 - Person and Dog are sublements of People.
+2 Jay Summet and Snoopy are text.
+2 age/classroom/breed are attributes and have correct values.
-1 for quotes in the wrong places or missing. -1 for \instead of forward slashes.

4. *(26 points)*
A table has been created for you with the following command:
CREATE TABLE baseball (Name TEXT NOT NULL, Game_Date TEXT NOT NULL, AtBats INT(10), Hits INT(10))

The database has contents such as the following (but with many more records):

The baseball table:

Name	Game_Date	AtBats	Hits
Babe Ruth	11/15/1930	5	2
Willie Mays	07/18/1965	10	6
Albert Pujols	10/17/2011	6	4

(a) [10 pts] Write a function called addStats that takes in a single parameter. The parameter will be a list of tuples, where the tuples are in the following form:

(Name, Date of Game, Number of At Bats, Number of Hits)

So a possible parameter passed into your addStats function could be:

[("Barry Bonds", "09/14/2007", 3, 1), ("Prince Fielder", "05/16/2010", 7, 5)]

The list can be of any length. Your addStats function should take the data from each tuple and insert it into the baseball table using PyMySQL code. The table is located on a DB server "exam4.gt.edu" , "Test4" is the database name, "cs2316" is the username, and "yellowjackets" is the password. You may assume every tuple within the passed in list will have four entries in the correct format necessary to insert into the baseball table. You may also assume the internet connection will always work (You do not need to do error checking).

Solution:

```
import pymysql

def addStats(stats):
   db = pymysql.connect(host = "exam4.gt.edu", user = "cs2316", passwd =  "yell
   for player in stats:
      sql = INSERT INTO baseball (Name, Game_Date, AtBats, Hits) VALUES (%s, %s

      cursor = db.cursor()
      cursor.execute(sql, player)
      cursor.close()
      db.commit()
   db.close()
```

Grading:
+1: imports pymysql
+1: has correct function header
+2: connects to database (+1 for a decent attempt, +1 for doing it correctly)
+1: Inserts all tuples into the database

+2: uses SQL statement (+1 for a decent attempt, +1 for doing it correctly)
+2: uses cursor (+1 for a decent attempt, +1 for doing it correctly)
+1: Closes DB and cursor(s).
-1 if they forget to commit after updating the database.

(b) [16 pts] Write a function called calcAverage that does not take in any parameters. Your calcAverage function should return a list of tuples, where the tuples are in the following form:

(Name of Player, Total Batting Average)

The returned list should have as many tuples as there are unique players in the baseball table above. The table can have any number of entries in it. Your calcAverage function should use PyMySQL code to select every distinct player and their total batting average, where:

$totalBattingAverage = TotalNumberOfHits/TotalNumberOfAtBats$

Realize that the total number of hits is not necessarily from one game that player played, but from every game that player has played; as well as for the total number of at bats. The database your function should connect to is the same one used in Part 1. Here, you may also assume the internet connection will always work. (Hint: If you cannot remember the SQL code that will automatically return the batting average per player for you, you can use a simple query that will return every row and then use python code to calculate the data.)

Solution:

```
EASY SOLUTION:
import pymysql
def calcAverage():
   db = pymysql.connect(host = ‘‘exam4.gt.edu’’, user = ‘‘cs2316’’, passwd = ‘

   sql = ‘‘SELECT Name, (SUM(Hits)/SUM(AtBats)) FROM baseball GROUP BY Name’’

   cursor = db.cursor()
   cursor.execute(sql)

   stats = cursor.fetchall()
   cursor.close()
   db.close()
   finalList = []
   for player in stats:
      finalList.append(player)

   return finalList
```

Grading:
+1: imports pymysql
+1: has correct function header
+2: connects to database (+1 for a decent attempt, +1 for doing it correctly)
+2: uses SQL statement (+1 for a decent attempt, +1 for doing it correctly)
+2: uses cursor (+1 for a decent attempt, +1 for doing it correctly)

+1: uses .fetchall() or the equivalent to (basically attempts to use retrieved data in python)
+2: appends to the final list (+1 for appending something, +1 for appending the correct tuple)
+3: Correctly finds total batting average (SQL or Python). (-2 for forgetting sum of hits/at bats, -1 for minor errors)
+1: returns the final list
+1: Closes the DB/Cursor

5. *(10 points)*
Review this textual XML file ("WinterIsHere.xml"):

```
<?xml version='1.0' encoding='UTF-8'?>
<GameOfThrones>
   <Westeros>
      <North>
         <Character Name = "John Snow">Winter is coming!</Character>
         <Character Name = "Hodor">Hodor</Character>
      </North>
      <Westerlands>
         <Character Name = "Joffrey Baratheon"/>
      </Westerlands>
   </Westeros>
   <Essos>
      <Braavos>
         <Character Name = "Arya Stark">Valar morghulis</Character>
      </Braavos>
      <Astapor>
         <Character Name = "Daenaerys Targaryan">Mother of dragons</Character>
      </Astapor>
   </Essos>
</GameOfThrones>
```

Write a function called **quoteFinder** that takes in no parameters. Your function should read in the xml (you can HARD CODE the xml file name into your code) and creates a dictionary, storing each Characters name as a key and their corresponding text as the value. For example: The Key/Value pairing for John Snow is, 'John Snow' : 'Winter is coming!' If there is no text associated with a character, their value will be None. The function should return the finalized dictionary.

Solution:

```
import xml.etree.ElementTree as etree

def quoteFinder():
    myDict = {}
    tree = etree.parse("WinterIsHere.xml")
    root = tree.getroot()

    for continent in root:
        for region in continent:
            for character in region:
                myDict[character.attrib['Name']] = character.text

    return(myDict)


Grading Rubric:
```

```
+1 for import xml.etree.ElementTree
+1 for correct def header
+1 for correct parsing of XML and use of getroot()
+2 for correctly iterating through children and accessing each individual character
+1 for correctly accessing characters name
+1 for correctly accessing characters text
+1 for correct handling of no text situation
+2 for correctly creating and returning finalized dictionary
```