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 will result in a substantial grade penalty.
  - 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 4 questions on 9 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	7	
3. Add Up Points	14	
4. Write XML	11	
Total:	41	

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] aliases

Solution: Two variables that point to the same data.

(b) [3 pts] keyword

Solution: A reserved word that is used by the compiler to parse program; you cannot use keywords (such as if, def, and while) as variable or function names (identifiers).

(c) [3 pts] semantic error

Solution: An error (in code) that leads to unexpected behavior. The program functions correctly (does what the code says) but the code does not actually perform the action that the programmer intended.

2. *(7 points)*
For each of the following multiple choice questions, indicate the single most correct answer by circling it!

(a) [1 pt] Which of the following is NOT a valid way to remove (only) the last item from aList?

A. `aList.pop()`

B. `aList = list(filter(lambda x: x != aList[len(aList)-1], aList))`

C. `aList = aList[0:-1]`

D. `aList = aList[0:len(aList)-1]`

(b) [1 pt] What is the correct syntax for adding a number (1) to a tuple?

A. `myTuple = myTuple + (1,)`

B. `myTuple.append(1)`

C. `myTuple.insert(0,1)`

D. `myTuple = myTuple + 1`

(c) [1 pt] Read the following code sample carefully.

```
import pymysql

db = pymysql.connect( host = 'sqlserver.example.com',
   passwd = 'SECRET12345', user = 'dbuser', db='myDatabase')
database = db
db = db.cursor()
sql = "INSERT INTO people (name, email, age) VALUES (%s, %s, %s)"
<BLANK>
db.close()
database.commit()
```

Which of the following choices when inserted into `<BLANK>` will successfully execute the query?

A. `cursor.execute( sql, ( "Jay", "jay@example.com", '77') )`
B. `cursor.execute( sql, "Jay", "jay@example.com", '77')`
C. `db.execute(sql, "Jay", "jay@example.com", '77')`
D. `db.execute( sql, ( "Jay", "jay@example.com", '77') )`
E. None of the above lines of code will execute the query.

(d) [1 pt] An SQL table was created as follows:

```
CREATE TABLE mytable (
year INTEGER,
month INTEGER,
day INTEGER,
hoursWorked INTEGER )
```

Which of the following queries will return the total hours worked in each month, in chronological order (earliest month first)?

A. `SELECT SUM(hoursWorked) FROM mytable ORDER BY year, month, day`
B. `SELECT SUM(hoursWorked) FROM mytable GROUP BY year, month ORDER BY year ASC, month ASC`
C. `SELECT SUM(hoursWorked) FROM mytable GROUP BY year, month, day`
D. `SELECT SUM(hoursWorked) FROM mytable GROUP BY year, month, day ORDER BY year, month, day`
E. `SELECT SUM(hoursWorked) FROM mytable GROUP BY year, month ORDER BY year DESC, month DESC`

Use the following code to answer the next two questions.

```
aList = [5, 10, 15, 20]
bList = 2 * aList
cList = bList
```

(e) [1 pt] What list does bList reference?

A. `[2, 5, 10, 15, 20]`

B. `[10, 20, 30, 40]`

C. `[5, 10, 15, 20, 5, 10, 15, 20]`

D. `[5, 10, 15, 20, 20, 15, 10, 5]`

(f) [1 pt] Which of the following statements is true?

A. cList is an alias of bList

B. bList is an alias of aList

C. cList is an alias of aList

D. cList is a copy of bList

(g) [1 pt] Suppose you want to extract all dates from a string of text, myText. The date will always be in the format YYYY-MM-DD. Which of the following will return a list of only these date strings?

A. `theDates = findall("[0-9]{4}.*[0-9]{2}.*[0-9]{2}", myText)`

B. `theDates = findall("\D{4}-\D{2}-\D{2}", myText)`

C. `theDates = findall("\d*-\d*-\d*", myText)`

D. `theDates = findall("\d{4}-\d{2}-\d{2}", myText)`

E. `theDates = findall("\S{4}-\S{2}-\S{2}", myText)`

3. *(14 points)*

You are given a CSV file with the names of students in a class and a score that represents how many points they earned in one day. The file is for multiple days, so a student name may appear more than once, with different number of points earned each day. The name is separated from the number of points earned with a comma. Here is a short example, but the real file can have any number of lines:

```
Will, 4
Alex, 18
Caleb, 3
Will, 2
Alex, 1
```

Write a function called **addUpPoints** that accepts a single string parameter representing the name of the file you are to open. Your function will return a dictionary where each entry is made up of a key consisting of the students name, and a value consisting of the sum of all the points they earned over all days as an integer. For example, the text file above would produce the dictionary:

```
{ 'Will': 6, 'Alex':19, 'Caleb': 3}
```

Solution: Using the CSV module:

```
import csv

def addUpPoints(filename):
    f=open(filename,'r')
    reader=csv.reader(f)
    lines=[]
    for line in reader:
        lines.append(line)
    f.close()

    students={}
    for student in lines:
        try:
            students[student[0]]=students[student[0]]+int(student[1])
        except:
            students[student[0]]=int(student[1])

    return students
```

```
#Not using CSV Module:

def addUpPoints(filename):
  f = open(filename)
  lines = f.readlines()
  f.close()
  students = {}
  for line in lines:
    data = line.split(",")
    name = data[0]
    number = int( data[1] )
    students[name] = students.get(name, 0) + number
  return students
```

Grading:
+1 for correct function header
+1 for opening the file in read mode
+2 for reading the data (via CSV or read or readlines or readline)
+1 for closing the file!
+2 for iterating through each line/record
+2 for extracting the name and number
+1 for converting the string number to an int.
+3 for inserting into dictionary properly (+1 for lookup, +1 for add, +1 for working the very first time)
+1 for returning a dictionary.

4. *(11 points)*
Review this textual XML file ("myFile.xml"):

```
<?xml version='1.0' encoding='UTF-8'?>
<Departments>
 <Department name="CS">
   <Class grade="A" name="CS 2316">
      <Section description="Awesome!" name="A1" />
   </Class>
 </Department>
 <Department name="ISYE">
   <Class grade="A" name="Optimization">I love this class!</Class>
 </Department>
</Departments>
```

Write python code that will create myFile.xml when it is executed using the appropriate xml module.

Solution:

```
import xml.etree.ElementTree as etree

root = etree.Element("Departments")
#Demonstrating SubElement style...
d1 = etree.SubElement(root, "Department", name="CS")
c1 = etree.SubElement(d1, "Class", name="CS 2316", grade="A")
s1 = etree.SubElement(c1, "Section", name="A1", description="Awesome!")

#Demonstrating apppending Element style..either style is acceptable.
d2 = etree.Element("Department", name="ISYE")
root.append(d2)
c2 = etree.Element("Class", name="Optimization", grade="A")
c2.text = "I love this class!"
d2.append(c2)

tree = etree.ElementTree(root)
tree.write("myFile.xml", "UTF-8")


Grading:
1pt for correct import
1pt for creating the root (Departments)
1pt for appending both DEPARTMENT elements under the root.
1pt for appending both Class elements under the correct departments
1pt for correctly appending the Section under the 2316 class.
```

```
2pts for getting all attributes correctly inserted in all the elements.
   (1pt if they get at least half the attributes correct.)
1pt for getting the TEXT on the optimization class correct.
1pt for creating the tree
2 pts for writing the tree to the write file and using UTF-8
```