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 4 questions on 7 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. Multiple Choice	4	
2. Read XML	9	
3. Code a Tree	12	
4. SeesNoAnts	6	
Total:	31	

1. *(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**

2. *(9 points)*
Review this textual XML file ("myFile.xml"):

```
<USA>
 <state population = "9700000">Georgia
   <city name = "Atlanta" population = "420000">
     <street population = "0">West Peachtree Street</street>
   </city>
   <city name = "Savannah" population = "140000"/>
 </state>
 <state population = "20000000">New York
   <city name = "New York" population = "8200000"/>
   <city name = "Buffalo" population = "260000"/>
 </state>
</USA>
```

And then review this python code that reads the xml file:

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

firstList = [ ]
secondList = [ ]
thirdList = [ ]
count1 = 0
count2 = 0
count3 = 0

found = root.findall("state")
for item in found:
   count1 = count1 + 1
   for item2 in item.find("city"):
       count2 = count2 + 1
       secondList.append(int(item2.attrib["population"]))
   for item3 in item.findall("city"):
       count3 = count3 + 1
       thirdList.append(float(item3.attrib["population"]))
   firstList.append(int(item.attrib["population"]))

a = (item.tag, count1)
b = (item2.tag, count2)
c = (item3.tag, count3)
myDict = {item.tag: min(firstList), item2.tag: min(secondList), item3.tag: min(thirdList)}
```

Write the contents of the following variables after the code is executed:

a =

b =

c =

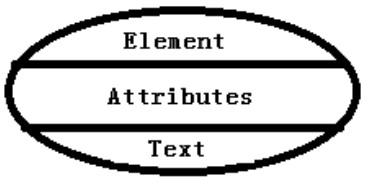
myDict =

Solution:

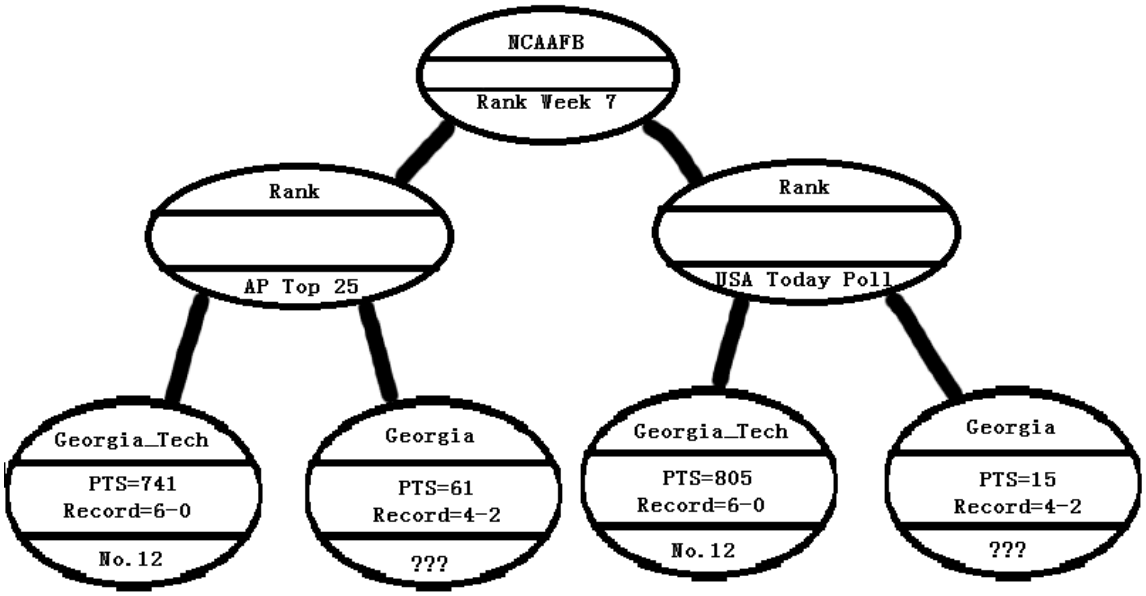
```
a = ("state", 2)
b = ("street", 1)
c = ("city", 4)

myDict = {"city": 140000.0, "state": 9700000, "street": 0}

Grading:
  +2 each for a,b,c
  +3 for myDict (only -1 if "city" value not a float.)
```

3. *(12 points)*

Given the XML tree below, write a funciton called powerRank which accepts a string as the only parameter. This parameter will specify the filename of the xml file you should write. Your function will create an XML file according to the tree, using the xml.etree.ElementTree module. Your code will "hard code" the data in the following tree, and DOES NOT need to work in other cases. Meaning, their won't be schools other than GT and UGA. The format of the tree is:



The data and structure of the tree is:



Solution:

```
import xml.etree.ElementTree as etree
def powerRank(filename):
    root=etree.Element("NCAAFB")
    root.text = "Rank Week 7"
    AP=etree.SubElement(root, "Rank")
    AP.text= "AP Top 25"
    GT1=etree.SubElement(AP, "Georgia_Tech", PTS="741", Record="6-0")
    GT1.text="No.12"
    UGA1=etree.SubElement(AP, "Georgia", PTS="61", Record="4-2")
    UGA1.text="???"

    US=etree.SubElement(root, "Rank")
    US.text="USA Today Poll"
    GT2=etree.SubElement(US, "Georgia_Tech", PTS="805", Record="6-0")
    GT2.text="No.12"
    UGA2=etree.SubElement(US, "Georgia", PTS="15", Record="4-2")
    UGA2.text="???"

    tree=etree.ElementTree(root)
    tree.write(filename, "UTF-8")
```

Grading:
Import the module correctly +1
Correct function header and parameter +1
Created the root with the given element name +1
Created 2 elements under the root with the given name and text. +2
Created 4 elements under the two rank sub-elements as pictures. +2
Placing data into the tag name, text, or attribute correctly on majority of elements +3.
Successfully write the XML into a file +2. (wrap up the tree +1, saved it as a file +1)

4. *(6 points)*

Write a function named `seesNoAnt` that takes in an antInfo tuple (described below) and returns a list of tuples that only includes the tuples for ants that do not have any other ants (enemy or friendly) in their view.

The antInfo tuple has the following format: `( (X1,Y1, View1), (X2,Y2, View2), ...)` Each inner antTuple has the X and Y position of an ant, followed by a tuple of what that ant can see. Each ant view (View1 and View2 above) is a tuple with 24 strings. (For this problem, the location that each entry in the ant view represents is unimportant.) Each string can be one of:

```
"" - empty string - empty square
"FH" - Friendly Anthill
"FA" - Friendly Ant
"W"  - A Wall
"EA" - Enemy Ant
"EH" - Enemy Hill
"F" - Food!
```

Your function should return a list of antTuples that contains all of the ants that have no other ants in their view (i.e. "EA" and "FA" do not appear in the view).

Solution:

```
def seesNoAnts(antinfo):
   goodants = []
   for ant in antinfo:
      view = ant[2]
      if "FA" not in view and "EA" not in view:
         goodants.append(ant)

   return goodants
```

Grading:
+1: Correct function header
+1: Covers case where ant sees EA
+1: Covers case where ant sees FA
+1: Considers all spaces in view
+1: Adds good ants to a list.
+1: Returns list of ant tuples