CS 2316
Exam 3 Fall 2011

Name : ______________________________

1. *(2 points)*
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 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
TA Name	2	
2. Multiple Choice	4	
3. Regular Expressions	8	
4. Classes	10	
5. Football Score	12	
6. Print Entries	15	
Total:	51	

2. *(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 is not valid:

A. `label.pack(sticky=W)`
B. `label.pack(side=TOP)`
C. `label.grid(row=1, columnspan=2)`
D. `label.grid()`

(b) [1 pt] Which of the following is true about buttons in Tkinter?

A. Buttons can respond to keyboard events as well as mouse events.
B. `b=Button(rootWin, command=self.clicked() )` will properly call the self.clicked method when the button is clicked.
C. Buttons must be created as instance variables.
D. An error is thrown if a command parameter is not included for a button when the button is created.

(c) [1 pt] What is the purpose of writing a class to contain GUI code?

A. It makes your code run faster.
B. It allows us to share references to our widgets easily between methods by making the widgets instance variables.
C. Widgets become globally accessible, no matter which function they are declared in.
D. It allows Python to delete all widgets easily when the window is closed.
E. It allows us to take advantage of polymorphism.

(d) [1 pt] When reading in CSV files using the CSV Reader module:

A. you do not have to manually close the file after you are finished.

B. you can use readlines to retrieve all of the rows from the file as a list of lists.

C. data is returned as a tuple of strings, where each string is one element in a row.

D. you need to iterate through the reader object to retrieve each row.

3. *(8 points)*

For each regular expression given below, determine which of the lines that follow it are completely matched by the regex pattern. (That is, there are no characters in the string that will not be matched by the given pattern...) Circle **all** lines that are fully matched.

(a) [2 pts] Pattern: `\$[0-9]{1,3}(,[0-9]{3})*`

123.456
$123
$123456
1,000,000
$12,456
$1,000,000.00

Solution: $123
$12,456
Grading: +2 for 100% correct. Minus 1 for each extra or missing circle, minimum score of zero on this part.

(b) [2 pts] Pattern: `(\d\d)*\d`

345
0000
abc
1234567
7778
1
A

Solution: 345
1234567
1
Grading: +2 for 100% correct. Minus 1 for each extra or missing circle, minimum score of zero on this part.

(c) [4 pts] Write a regular expression that will match a string formatted as: "City, ST" where city is a single word made up of only letters (uppercase or lowercase), followed by a comma and space, then exactly two upper case letters. You may assume that the city name has at least one letter. For example, the strings:
Marietta, GA
and
Notacity, ST

Solution: One possible solution:
`[a-zA-Z]+, [A-Z]{2}`
Grading: +1 for matching one or more letters for the city
+1 for matching the comma and space +1 for matching uppercase letters for the ST
+1 for matching exactly 2 letters for the ST.

4. *(10 points)*
You are given a table named assignments in an SQL database. The table has the following columns:

id: INTEGER type, auto-increment, primary key
name: TEXT type (Holds the name of the assignment as text)
totalpoints: INTEGER type (Holds the total worth of the assignment
category: TEXT type (Holds the category to which the assignment belongs)
duedate: TEXT type (Holds the date the assignment is due, in MM/DD/YYYY format)

Write the SQL queries described below. Write only the SQL query; do not write any Python code.

(a) [2 pts] Retrieve all data from assignments where the name of the assignment is "Exam 3".

Solution: SELECT * FROM assignments WHERE name="Exam3"
+2 if 100% correct, +1 if mostly correct.

(b) [2 pts] Retrieve the due date (only) for all assignments with "Exam" anywhere in the name.

Solution: SELECT duedate FROM assignments WHERE name LIKE "%EXAM%"
+2 for 100% correct, +1 if mostly correct.

(c) [2 pts] Set the due date of all assignments in the "Homework" category to be "11/5/2011".

Solution: UPDATE assignments SET duedate="11/5/2011" WHERE category = "Homework"
+2 for 100% correct, +1 if mostly correct.

(d) [2 pts] Remove any record where the total points is less than 5.

Solution: DELETE FROM assignments WHERE totalpoints < 5
+2 for 100% correct, +1 if mostly correct.

(e) [2 pts] Insert a new record with only a name and category, where the name is "TL3" and the category is "Exams".

Solution: INSERT INTO assignments (name, category) VALUES ("TL3", "Exams")
+2 for 100% correct, +1 if mostly corect.

5. *(12 points)*
A website at http://www.gtscores.com/latest.html has the following format:

```
<html>
<title>Georgia Tech Football</title>
<table>
<tr>
<td title="Date">10/29/11</td>
<td title="Opponent">vs. Clemson</td>
<td title="Location">Atlanta, Ga.</td>
<td title="Score">W, 31-17</td>
</tr>
</table>
</html>
```

Write a function called **findScore** which accepts no parameters. Your funciton will go to that URL (hard code the url in your code) and download the webpage. Then it should extract the two scores (31 and 17 in this example) and return them as a tuple of integers. Note that the score can be single digits, or on a very good day, 3 digits. You may assume that the table will only contain information about the last game, as in the example above.

Example run:

```
>>> scores = findScore()
>>> print(scores)
(31,17)
```

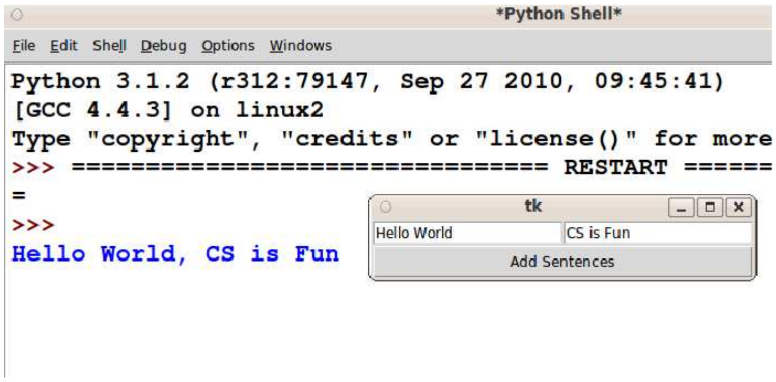
Solution:

```
def findScore():
import urllib.request
response = urllib.request.urlopen('http://www.gtscores.com/latest.html')
html = response.read()
htmlString = str(html)
whereDash = htmlString.find("-")
winningStart = whereDash
losingEnd = whereDash
while htmlString[winningStart] != " ":
winningStart -= 1
while htmlString[losingEnd] != "<":
losingEnd += 1
```

```
winScore = htmlString[winningStart+1:whereDash]
loseScore = htmlString[whereDash+1:losingEnd]
return ( int(winScore), int(loseScore)  )
```

Grading: +1 valid function header
+1 imports urllib.request
+1 correct urlopen
+1 response.read()
+1 casts response to string
+1 returns tuple
+1 returns tuple of integers
Finding the actual scores:
+2 looks for an identifying characteristic
+3 gets correct information (+1 if only one correct score)

6. *(15 points)*
Create a Graphical User Interface (GUI) that contains two entry widgets and a single button. The entry widgets are both on the first line, with the button below them that streches to fit the window (HINT: using the grid layout might make this easier). When the user clicks the button, the GUI program must take the sentences stored in the entries, combine them with a comma and space in-between, and print out the result. For example, when entry 1 contains "Hello World" and entry 2 contains "CS is fun", then clicking the button would result in the following print out: "Hello World, CS is fun". (to the console)



```
Python 3.1.2 (r312:79147, Sep 27 2010, 09:45:41)
[GCC 4.4.3] on linux2
Type "copyright", "credits" or "license()" for more
>>> ================================ RESTART ======
=
>>>
Hello World, CS is Fun
```

Hello World | CS is Fun

Solution:

```
from tkinter import *
class GUI:
def __init__(self, win):
self.entry1 = Entry(win)
self.entry1.grid(row = 0, column = 0)

self.entry2 = Entry(win)
self.entry2.grid(row = 0, column = 1)

Button(win, command=self.clicked, text="Add Sentences").grid(row = 1, column = 0, c


def clicked(self):
sentence1 = self.entry1.get()
sentence2 = self.entry2.get()
print(sentence1 + ", " + sentence2)

root = Tk()
GUI(root)
root.mainloop()
```

Grading:
+3 starting TK, instantiating object, running mainloop (+1 per line)
+2 creates entry 1 on line 1 (+1 for create, +1 for proper location)
+2 creates entry 2 at line 1 (+1 for create, +1 for proper location)
+2 create button under the entries
+2 button is gridded/formated correcly (+1 sticky, +1 columnspan)
+1 button is binded to a function
+1 gets entry 1 text
+1 gets entry 2 text
+1 prints correct sentence (this is all or nothing)