CS 2316

Exam 2 Fall 2013

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 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	6	
2. Regular Expressions	9	
3. Phone Bill	14	
4. Survey GUI	21	
5. countSmiles	9	
Total:	59	

1. *(6 points)*
   For each of the following vocabulary terms, write a concise 1-2 sentence definition. Be brief, and to the point.

   (a) [3 pts] object

   Solution: A compound data type that is often used to model a thing or concept in the real world. It bundles together the data and the operations that are relevant for that kind of data. Instance and object are used interchangeably. (An object whose type is of some class.)

   Looking for: Compound Data Type, Includes data & operations/functions/methods.

   (b) [3 pts] class

   Solution: A user-defined compound type. A class can also be thought of as a template for the objects that are instances of it.

   Looking for: Compound Data Type , can specify functions and group data, used to instantiate objects.

2. *(9 points)*
   For each regular expression given below, write a string that it matches.

   (a) [1 pt] Pattern: `CS 2316`

   Solution: CS 2316 exactly. Grading: +1 for 100% correct, zero otherwise.

   (b) [2 pts] Pattern: `[A-Z]{2,4} \d\d\d\d`

   Solution: Anything that has 2 to 4 letters followed by a space and 4 digits, e.g. CS 2316 Grading: +2 for 100% correct. Minus 1 for each extra item.

   (c) [2 pts] Pattern: `\d\d[5-9]{2}\s[A-Za-z]+\s?[A-Za-z]*`

Solution: Looks like a street address. Must have 4 digits with the last two only in the range 5-9, a space, and then one word street name or two word street name. Example: 1359 Miller Way or 0865 Atlantic Grading: +2 for 100% correct. Minus 1 for each extra/missing item.

(d) [2 pts] Pattern: `[^<>@!]+@[a-z]+\.com`

Solution: Looks like an email address. The "username" part can have any letters OTHER than a `< > ! or @`. The @ is followed by only lower case letters and then a .com. Example: will.barr@fdas.com

Grading: +2 for 100% correct. Minus 1 for each extra/missing item.

(e) [2 pts] Pattern: `[a-z]-[A-Z]...`

Solution: Any small letter, followed by a dash, followed by any large letter, followed by any three characters.

Example: z-A!!!

Grading: +2 for 100% correct. Minus 1 for each extra/missing item.

3. *(14 points)*
Write a function named **phoneBill** that takes in a list of tuples. Each tuple will consist of a name, the number of text messages, and the number of minutes that a particular person has used.

Your function should return a list of tuples that consist of a name and the dolar amount of that person's phone bill as a floating point number.

All users are on the same plan which costs $25 a month, and includes 500 texts and 300 minutes. If users go over their limits, they are charged $0.10 for each additional text message (over 500). For minutes 301-400 (the first 100 minutes over) users are charged $0.10 a minutes. For minutes 401- and up they are charged $0.25 a minute.

Example test case:

```
>>> result =  phoneBill( [ ('Bryan',500,300), ('Jay',501,200), ('Will',0,301),
      ('Jarrod',510,310), ('Nolan',100,410),('Alex',550,450) ]  )
>>> result
[ ('Bryan', 25.0), ('Jay', 25.1), ('Will', 25.1),
      ('Jarrod', 27.0), ('Nolan', 37.5), ('Alex', 52.5) ]
```

Solution:

```
def phoneBill(userData):
    billData=[]
    for user in userData:
        name=user[0]
        texts=user[1]
        mins=user[2]
        bill = 25.0
        if texts > 500:
           bill = bill + 0.1 * (texts-500)

        if mins > 400:
           bill = bill + (0.1 * 100) + (mins-400) * 0.25
        elif mins > 300:
           bill = bill + 0.1 * (mins-300)

        billData.append( (name,bill) )
    return billData
```

Grading: 1 point for a correct header.
3 points for extracting the name/texts/minutes numbers correctly.

1 point for charging everybody the base $25
2 points for charging 0.10 for all text messages ABOVE 500
2 points for charging 0.10 for all minutes in the 301-400 range.
2 points for charging 0.25 for all minutes in the 401-max range.
1 point for creating a tuple with (name,bill).
1 point for appending to a list.
1 point for returning the list of tuples.

4. *(21 points)*
You want to create a GUI program that will parse a simple CSV data file to determine if Georgia Tech will first renovate Skiles or the Instructional Center. The survey data is stored as a CSV file which will need to be opened and read.

The CSV file will have at least one line, and can be of any length. The data will be stored in the format:
Name, Skiles Score (0-10), IC Score (0-10)
EXAMPLE:

```
Josh, 9, 3
Jake, 2, 10
```

Your program should calculate the sum of all Skiles scores and IC scores by adding up the data from all lines in the file. Your program should create a GUI that has the following elements:

A text entry, for the user to enter the name of the CSV file into. This text entry field should labeled "File Name:" using a label next to it.

Two "readonly" entries to display the result for the IC and Skiles. Each of these entries should also be labeled with labels ("Skiles Score", and "IC Score").

A large button "Get Results" that spans the entire bottom of the GUI.

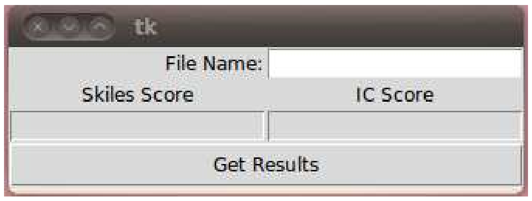
Behavior:

When the user clicks the "GetResults" button, your program should check to see if a file name has been entered into the first entry. If not, your code should pop up an info dialog messages (Title: "Info", Text:"Please enter a file name first!") and do nothing else.

If a file name exists in the first entry, your program should attempt to open the CSV file named. If you are unable to open the file your code should pop up an error dialog message (Title: "Error", Text:"CSV File problem!") and do nothing else. (If you ARE able to open the file, you may assume it has valid data.)

Assuming the file was opened successfully, read the data in and sum up the scores for Skiles and the scores for the I.C. and place the numbers into the two "read-only" entries.

Your GUI should have the various elements in the same general position as this example:



Note that the "File Name:" label is directly to the left of its entry, while the IC Score and Skiles Score labels are centered above their entries. Also note that the "Get Results" button spans the entire width of the bottom of the window.

Please please your answer on the following page:

This page left blank for answer:

Solution:

```
from tkinter import *
import csv

class survey:
    def __init__(self, win):
        self.fileName=StringVar()
        self.SkilesScore = StringVar()
        self.ICScore = StringVar()

        Entry(win, textvariable=self.fileName).grid(row=0,column=1)
        Label(win, text="File Name:").grid(row=0,column=0, sticky=E)


        Label(win, text="Skiles Score").grid(row=1,column=0)
        e1 = Entry(win, textvariable=self.SkilesScore, state="readonly")
        e1.grid(row=2,column=0)

        Label(win, text="IC Score").grid(row=1, column=1)
        e2=Entry(win, textvariable=self.ICScore, state="readonly")
        e2.grid(row=2,column=1)

        b = Button(win,text='Get Results',command=self.clicked)
        b.grid(row=3,column=0,columnspan=2,sticky=EW)

    def clicked(self):
        if self.fileName.get()=="":
            messagebox.showinfo("Info", 'Please enter a file name first!')
            return
        try:
            f=open(self.fileName.get(), 'r')
            lines=[]
            reader=csv.reader(f)
            for row in reader:
                lines.append(row)
            f.close()
        except:
            messagebox.showwarning("Error!", 'CSV File Problem')
            return
        skiles=0
        ic=0

        for thing in lines:
            skiles=skiles+int(thing[1])
            ic=ic+int(thing[2])

        self.SkilesScore.set(skiles)
        self.ICScore.set(ic)

win=Tk()
app = survey(win)
win.mainloop()
```

Grading:
+1 point for importing tkinter
+3 for creating GUI (Tk() call and Object creation) and running mainloop
+3 points for correct labels and position.
+3 points for correct entries and position. (-1 if outputs are not readonly!)
+2 points for GetResults button (which spans the window).
+1 point for correctly displaying Info dialog when FileName entry is empty.
+1 point for correctly displaying Error dialog on file open exceptions.
+3 for correctly opening and reading the CSV data.
+2 for correctly summing up the results.
+2 for putting the numbers into the correct place on the GUI.

5. *(9 points)*
You are to write a function named countSmiles which will accept a string representing the URL of a website. Your objective is to download the HTML from this website and return an integer representing the number of times a smiley occurs. A Smiley is the two character combination of a colon and close parenthesis :).

Solution:

```
import urllib.request
from re import findall

def countSmiles(website):
    response = urllib.request.urlopen(website)
    html = str(response.read())
    data = findall(":\)", html)
    return len(data)
// Or, if they didn't use regular expressions, they could do this:
   counter = 0
   index = html.find(":)")
   while index != -1:
      index = html.find(":)",index+1)
      counter = counter+1
   return counter
```

Grading:
1 point for correctly importing urllib
3 points for correctly downloading the HTML.
4 points for counting the smiles (using re's or otherwise) (+3 if their code almost works, +2 if it would work with minor fixes, +1 if they have the right idea but the code is horribly wrong.)
1 point for returning an integer