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 6 questions on 11 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. Fill a Dictionary	3	
3. Point Objects	5	
4. GUI drawing	12	
5. print Pi	10	
6. Rushing Yards	15	
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] 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.

   (b) [3 pts] slice

   Solution: A subsequence copied from a sequence specified by a range of indices. The slice operator is: sequence[start:stop].

   (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. *(3 points)*
   Examine the code below. Write the contents of the dictionary after the code runs. You may either list a table of key/value pairs, or write what would be printed to the screen if you executed `print(myD)` at the python prompt.

```
myD = {}

for i in range(5):
   myD[i] = i*20
```

Solution:

```
Key Value
0 0
1 20
2 40
3 60
4 80
```

Grading:
3pts = table completely correct. (They can also write out a dictionary using python syntax if they get it correct...)
2 pts - Forget one table entry.
1 pt - Get just the keys or just the values.

3. *(5 points)*
Examine the following class definition of a Point and the seven lines of code that use Points:

```
class Point:
    def __init__(self,x=0,y=0):
        self.x = x
        self.y = y

    def getDistance(self):
        from math import sqrt
        dist = sqrt( self.x*self.x + self.y*self.y)
        return(dist)

    def __eq__(self,other):
        if self.x == other.x:
            return True
        else:
            return False

p1 = Point()
p2 = Point(3,4)
p3 = Point(3,3)

v1 = p1.x
v2 = p2.y
v3 = p2 == p3
v4 = p2.getDistance()
```

What are the values in the 4 variables (v1,v2,v3,v4) after this code executes?

Solution: v1: 0 (accept 0.0 but don't give bonus point)
v2: 4 (accept 4.0 but don't give bonus point)
v3: True
v4: 5.0 (accept 5 but don't give bonus point)

Grading:
+1 for each correct answer. +1 bonus if all 4 answers are correct (including correct float/int)

4. *(12 points)*
Examine the following GUI code:

```
from tkinter import *

class GuiTest:
    def __init__(self,rootWin):
        frame=Frame(rootWin)
        frame.pack()
        self.button=Button(frame,text="Add Star!",command=self.clicked)
        self.button.pack()
        self.entry1=Entry(frame,width=60,state=NORMAL)
        self.entry1.insert(0,"***")
        self.entry1.config(state="readonly")
        self.entry1.pack(anchor=W)
        self.label1=Label(frame,text="Number of Stars:")
        self.label1.pack()
        self.entry2=Entry(frame,text="3")
        self.entry2.config(state="readonly")
        self.entry2.pack(anchor=E)

    def clicked(self):
        len1=len(self.entry1.get())
        len1 = len1 + 1
        if len1>5:
            len1=0
        entry= '*' * len1
        self.entry1.config(state="normal")
        self.entry1.delete(0,END)
        self.entry1.insert(0,entry)
        self.entry1.config(state="readonly")

rootWin=Tk()
rootWin.title("2316 Tester")
app=GuiTest(rootWin)
rootWin.mainloop()
```

Draw exactly what is rendered on the screen when the code above is executed and after the button is clicked 7 times! Make sure to include any and all window decorations.

Solution:

Grading:

```
+2 Window Decorations
+1 At least has 1/3 of minimize, maximize, close buttons
+1 Correct title of "2316 Tester"
+10 GUI Elements
+1 Button with correct text
        +1 button in correct location
+1 Text entry box in middle
+1 Text entry box in middle has width 60 (clearly larger than any other element)
+1 Label in correct place w/ correct text.
+1 Entry on bottom
+1 Entry on bottom is on right side
+1 Entry on bottom does not have any text
+2 Entry in middle has 4 stars.
```

This page intentionally left blank. You may use it as scratch paper. If you place an answer on this page, box it, label it clearly, and indicate clearly on the original problem page that your answer is on this page.

5. *(10 points)*

Write a function called `printPi` that accepts a list (of lists) as its parameter. This list (of lists) will represent the number Pi in a table such that the numbers flow VERTICALLY! Here is an example of sample input that has Pi represented to 99 digits. (You may assume that each row is the same length, but your code must work with tables of data that are smaller or larger than this example.)

```
aList = [[3, 3, 4, 7, 7, 1, 4, 6, 9, 2], ['.', 5, 6, 9, 1, 0, 4, 4, 9, 5],
  [1, 8, 2, 5, 6, 5, 5, 0, 8, 3], [4, 9, 6, 0, 9, 8, 9, 6, 6, 4],
  [1, 7, 4, 2, 3, 2, 2, 2, 2, 2], [5, 9, 3, 8, 9, 0, 3, 8, 8, 1],
  [9, 3, 3, 8, 9, 9, 0, 6, 0, 1], [2, 2, 8, 4, 3, 7, 7, 2, 3, 7],
  [6, 3, 3, 1, 7, 4, 8, 0, 4, 0], [5, 8, 2, 9, 5, 9, 1, 8, 8, 6] ]
```

To convince yourself of this, compare the first 10 characters of pi with the first column: 3.14159265

Your printPi function should print the digits of Pi out in order, on a single line.

Solution:

```
def printPi(theList):
   for col in range(0,len(theList[0]) ):
      for row in range(0, len(theList) ):
         print(str(theList[row][col]), end="")
```

Grading:
+2 method header (def and name, plus parameter) +2 iterates through columns +2 and then rows +1 indexes into the correct value each time. +1 prints answer without line breaks +1 prints correct answer. +1 Doesn't return a value

6. *(15 points)*

Write a function named `leadingRushYards` that doesn't take in any parameters. The function should open a file named data.csv in the current working directory, which contains data formatted as follows:

TeamName1, Rushing yards in 1st game played, rushing yards in 2nd game played, etc...

For example, an excerpt from the table might look like this:

```
Georgia Tech, 297, 382, 604, 312, 296
UGA, 137, 188, 194, 207, 155
Oklahoma, 246, 111, 144, 208
```

Your function should calculate the average rushing yards per game for each team (in other words, the average of all the values for that particular row, excluding the team name). It should return a tuple containing the highest rushing yards average, as well as the name of the team that has that average: (AvgYards, TeamName), with the team name as a string and the average yards as a float. Notice that the number of data points for any given team is unspecified and not necessarily the same for all teams: in the above example, GT and UGA both have 5 games played, but Oklahoma only has 4; other teams in the same data file might have different numbers of games played as well. You may assume that any team in the file will have played at least one game.

HINT: If you sort a list of tuples, the tuples will be sorted based on the first value in the tuples (then by the 2nd value in the tuple if there is a tie)

Example output for the data above:

```
>> res = leadingRushYards()
>>res
(378.2, "Georgia Tech")
```

Solution:

```
import csv

def leadingRushYards():
    file = open("data.csv")
    reader = csv.reader(file)

    data = []
    for item in reader:
        floats=[]
        floats.append(item[0])
```

```
        for thing in item[1:]:
            floats.append(float(thing))
        data.append(floats)

    newData=[]
    for item in data:
        newData.append( ( sum(item[1:]) / len(item[1:]), item[0])  )
    file.close()
    return max(newData)
```

Grading:
+1: funtion header correct
+1: imports csv or correctly uses open()
+1: opens the file
+1: creates csv.reader object or splits on commans if manually opening
+1: closes the file
+2: extracts data to new list
+2: Properly converts the appropriate values to floats
+2: Correctly finds the avg yards for a team
+2: Correctly finds the max avg
+1: Correctly matches max avg to the team name.
+1: Returns a tuple

This page intentionally left blank. You may use it as scratch paper. If you place an answer on this page, box it, label it clearly, and indicate clearly on the original problem page that your answer is on the last page.