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 11 questions on 16 pages including the title page. Please check to make sure all pages are included. You will have 100 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. Classes and Objects	5	
2. Multiple Choice	6	
3. List Questions	6	
4. iPhone	5	
5. Bravarian Strings	5	
6. GUI drawing	10	
7. Find Last Period	8	
8. GetTemp	10	
9. QB Rating	8	
10. Extract 21	12	
11. countSmiles	9	
Total:	84	

1. *(5 points)*
Define **class** and **object**. Explain how they are related. Explain their similarities and their differences. Be brief, and to the point.

Solution: Example solution:

A class is a user-defined compound data type that encapsulates a grouping of related data and the behavior/functions/code that are relevant to the data. It is a template that tells python how to create an object and how the object behaves.

An object is an instance of a class. It is the actual location in memory that holds the data for a specific instance of a class. You can create multiple objects from the same class. They will share the same behavior (methods) but may have different state (data).

Grading: +1 point for each of the following items, 5 max
class is a template for creating objects (defined with code)
class specifies grouping of data and behavior (user defined CDT)
Objects are "made from" classes (multiple objects from same class)
Different objects of the same class will share the same behavior
Different objects of the same class may have different data/state
Classes can be used to represent a "real world" concept/idea (e.g. for simulations)

2. *(6 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 statements about Radiobuttons are false?

A. Radiobuttons have a 'state' parameter

B. Radiobuttons have a 'variable' parameter

C. Radiobuttons have a 'value' parameter

D. Radiobuttons can change the value of StringVar's

E. All of the above statements are true.

(b) [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)`

Use the following code to answer the next two questions.

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

(c) [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]`

(d) [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

(e) [1 pt] Assume the class Test1 creates a GUI and has been defined correctly. Examine the following lines of code:

```
rootWin = Tk()
question = Test1(rootWin)
```

Which of the following lines of code, if executed after the above code, will cause Python to throw an error?

A. `Test1.difficulty = "moderate"`

B. `experience = Test1(rootWin)`

C. `rootWin.mainloop()`

D. `app.Label(rootWin, text = "LeBron")`

E. `question.answer = "here?"`

(f) [1 pt] Which of the following data types are NOT sequences?

A. Tuples **B. Dictionaries** C. Strings D. Lists E. All of these are sequences

3. *(6 points)*
Examine the following code:

```
a=["Apache","CS2316","Python",["Bacon","Toast","Eggs","Avocado"]]
b=a[-1]
c=a.reverse()
d=b.sort()
e=a[-1]
f=a[0]
```

After the code executes, what does each of the variables point at?

a -

b -

c -

d -

e -

f -

Solution:

```
a=[['Avocado', 'Bacon', 'Eggs', 'Toast'], 'Python', 'CS2316', 'Apache']
b=['Avocado', 'Bacon', 'Eggs', 'Toast']
c=None
d=None
e='Apache'
f=['Avocado', 'Bacon', 'Eggs', 'Toast']
```

Grading: +1 point for each correct answer.

4. *(5 points)*
Examine the following code and write down exactly what is printed to the screen when it is executed:

```
class iPhone:
    def __init__(self,software):
        self.software=software
        self.color="pink"

    def printsoftware(self):
        print("My iPhone's software is",self.software)
```

```
    def changePhones(self,phone):
        self.printsoftware()
        self.phone=phone
        self.phone="Android"
        return print("My phone is "+self.phone)

myPhone=iPhone("6.1.4")
print(myPhone.color)
print(myPhone.changePhones("Windows"))
```

Solution:

```
pink
My iPhone's software is 6.1.4
My phone is Android
None
```

Grading:

+1 for each correct line
+1 if all 4 lines are correct with no extra lines.
-1 for any extra lines.

5. *(5 points)*
Examine the following code.

```
def iSeeYou():

    aString = "THIS IS NOT A TEST"
    bString= aString[::-1]
    aLetter = bString[12]
    bLetter = bString[0]

    cString = "MULL IT OVER"
    dString = cString[3:0:-1]

    nonsenseString = "MLOIOWNCXA"
    eString = nonsenseString
    fString = eString[0:7:3]
```

```
    print(aLetter + dString + fString + nonsenseString[9]+ bLetter + aLetter)

iSeeYou()
```

What is printed to the screen when this code is executed?

Solution:

```
ILLUMINATI
```

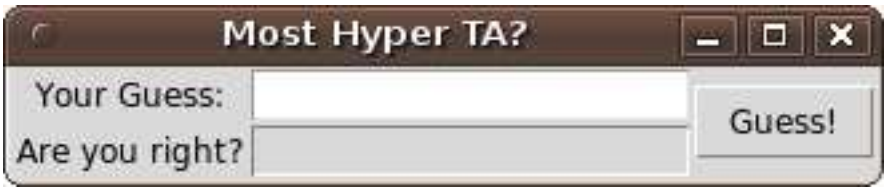
Grading: 1 point for "I" in front and back
1 point "LLU"
1 point for "MINA"
1 point for "T".
1 point for getting the full word correctly
-1 point for any "extra" incorrect letter(s) between correct segments.

6. *(10 points)*
Given the following code, draw the GUI that is produced TWICE. Draw it once as it first appears. To the right of that, draw the GUI a second time after the word Samantha is typed into the self.entry1 entry and the button is pressed. Include the window with any decorations. Indicate colors, shading, or state with arrows and labels.

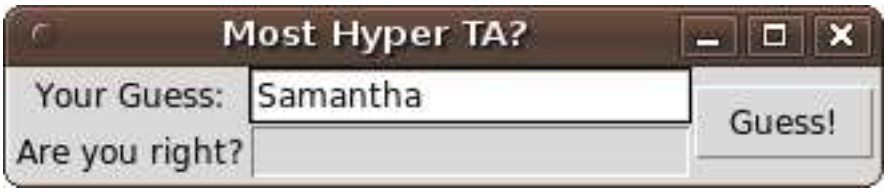
```
from tkinter import *
class MyScreen:
    def __init__(self,Window):
        self.var=StringVar()
        button1 = Button(Window, text="Guess!", command=self.clicked)
        button1.pack(side=RIGHT)
        frame = Frame(Window)
        frame.pack(side=RIGHT)
        label1 = Label(frame, text="Your Guess:")
        label1.grid(row=1,column=2)
        self.SV = StringVar()
        self.entry1 = Entry(frame, textvariable= self.SV)
        self.entry1.grid(row=1, column=4)
        label2=Label(frame, text="Are you right?")
        label2.grid(row=2,column=2)
        self.entry2 = Entry(frame)
        self.entry2.grid(row=2, column=4)
        self.entry2.config(state="readonly")


    def clicked(self):
        self.entry2.delete(0, END)
        if self.entry1.get()=="Samantha":
            self.entry2.insert(0, "Correct!")
        else:
            self.entry2.insert(0, "WRONG")

Window = Tk()
Window.title("Most Hyper TA?")
app = MyScreen(Window)
Window.mainloop()
```

Solution:

As appears:



After text entry and button Press (entry is read-only):



Grading:
Before button press:
+1 window correctly drawn with decorations
+1 window title correct
+1 Button on the far right and centered vertically.
+1 "Your Guess" above "Are you right?".
+1 Labels to the left of entries.
+1 Bottom Entry indicated as read-only (label/shading)
+2 Empty rows/columns don't take up space. (e.g. no row 0, col 0,1,3)

After button press:
+1 "Samantha" in top entry.
+1 No text in bottom entry.
-1 for any other changes.

7. *(8 points)*

You need to extract the file extension from a filename such as "GeorgeP.Burdell.gif". To do this, you need to search through a string to find the position (index) of the LAST period. Write a function named **findLastPeriod** that takes a string as a parameter and returns the position of the last period in the string as an integer. If the string has no periods, return -1.

Example use case:

```
>>> x = findLastPeriod("test.txt")
>>> print(x)
4
```

Solution:

```
def findLastPeriod(myStr):
    for index in range(len(myStr)-1,-1,-1):
       if myStr[index] == ".":
          return index
    return -1

alternate solution:
def findLastPeriod(myStr):
  index = len(myStr) - 1
  while index >= 0:
     if myStr[index] == ".":
        return index
     else:
        index = index -1

  return -1   #or index....
```

Grading:
1 point for correct header.
2 points for iterating through most of the characters in the string
1 point for including end of the string (and not one past!)
1 point for including the beginning of the string (and not stopping at 1!)
2 points for returning the index of the LAST period in the string.
1 point for returning -1 if the string has no periods.

8. *(10 points)*

You are hired to write the control software for a bio-reactor. As part of this job, you need to write a function named `getTemp` that takes no parameters. It should display a message (prompt) to the user asking them to enter a temperature between 30 and 90 degrees C. (inclusive) ("Please enter a temp between 30 and 90 degrees C") Note that the user may enter a temp such as 45.8 which is valid.

Your function should return the temperature the user entered as a float. If the user does not enter a valid temperature (not a number, lower than 30 or higher than 90) you should print out "Invalid Temp, try again!", and then repeat the prompt asking for a temperature until they get it right.

Solution:

```
def getTemp():
   userStr = input("Please enter a temp between 30 and 90 degrees C")
   try:
     userFlt = float(userStr)
     if userFlt > 90 or userFlt < 30:
        print("Invalid Temp, try again!")
        return getTemp()
     else:
        return userFlt
   except:
     print("Invalid Temp, try again! ")
     return getTemp()
```

Note: Students could also use a while loop instead of recursion to solve this problem.

Grading:
1 point for prompting the user with the correct text
1 point for trying to convert to a float.
2 points for correctly catching float conversion exceptions
2 points for correctly checking >=30 and <=90. (-1 for not allowing 30 or 90)
2 points for asking again and again until the user gets it right.
2 point for returning the correct float value.

9. *(8 points)*
Write a function called `qbRating` that accepts a dictionary as a parameter. The dictionary will be structured as follows:

```
nfl = { 'Falcons':[360,'Matt Ryan'], 'Broncos' : [450, 'Peyton Manning'],  ...}
```

such that each key/value pair is of the form: `team : [passing yards, "Players Name"]` The dictionary may have the player with the top passing yardage from many different teams, not just the 2 shown in our example. The qbRating function will return a list of tuples, in the following form:

```
[ (passing yards, "Player Name"), (passing yards, "Player Name"), ...]
```

The tuples must be sorted by the number of passing yards (largest first). For the example above, the output would look like: `[ (450, 'Peyton Manning'), (360, 'Matt Ryan'), ... ]`

Solution:

```
def qbRating(adict):
    new = []
    for i in adict:
        listy = adict[i]
        theTuple = (listy[0],listy[1])
        new.append(theTuple)
    new.sort() # or new.sort(reverse=True) and not new[::-1] below...
    return new[::-1] #Reverse the list before returning.
```

Grading:
+1 correct function header
+2 iterates through the dictionary
+2 correctly adds a tuple of values to a new list.
+2 sorts the final list.(+1 for any sort, +1 for correct order)
+1 returns the correct list

10. *(12 points)*
Write a function `extract21` that takes in one parameter, the name of a CSV file, as a string. This CSV file contains the names and ages of a group of students. (lastname, firstname, age). Open the file, read in the data, and extract only those students who are 21 or over. Sort the data by lastname (then firstname, then age to break ties). Write the extracted data to a file called "over21.csv" using the same (lastname, firstname, age) format.

For example, if our input file looked like this:

```
Kassem, Samantha, 20
Ferenzio, Bobbay, 18
Stein, Jarrod, 21
Avery, Jordan, 21
Smith, Scotty, 20
Lawler, Harry, 22
```

the output file should look like this:

```
Avery, Jordan, 21
Lawler, Harry, 22
Stein, Jarrod, 21
```

Solution: Example solution not using the csv module:

```
def extract21(fileName):
    readFile=open(fileName)
    nestedList=[]
    for i in readFile.readlines():
        splitUp=i.split(",")
        nestedList.append([splitUp[0],splitUp[1],int(splitUp[2])])
    nestedListOver21=[]
    for i in nestedList:
        if i[2] >=21:
            nestedListOver21.append(i)
    nestedListOver21.sort()
    newFile=open("over21.txt","w")
    for i in nestedListOver21:
        newFile.write(i[0]+", "+i[1]+", "+str(i[2])+"\n")
    newFile.close()
    readFile.close()
```

Grading:
+1 Correct function definition
+1 Opens the file
Using CSV Reader:
+1: Imports CSV
+1: Correctly creates CSV reader
+1: Iterates through the reader/gets rows out.
Not using CSV Reader:
+2: Gets each line from the file.
+1: Splits the line on commas

+2: Converts 3rd item to an int or float
+2: Compares each age to 21 and extracts data
+1: Sorts the output data correctly
+1: writes output file correctly.
+1: Closes the output file

11. *(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

This page intentionally left blank. You may use it for scratch paper. If you place an answer on this page, box it, indicate which problem it is for by number, and BE SURE TO WRITE "Answer on last page" at the problem location!