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 8 questions on 10 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. Multiple Choice	11	
3. DooWaa	6	
4. List Questions	4	
5. Alias vs. Copy	6	
6. Count Capitals	8	
7. Find Last Period	8	
8. Rank Days	10	
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] exception

Solution: An error that occurs at runtime. Can be handled using the try/except construct.

(b) [3 pts] increment

Solution: increment - The process of increasing a variable, typically by one. aVar = aVar + 1

2. *(11 points)*
For each of the following multiple choice questions, indicate the best correct answer. Indicate your selected answer by circling it.

(a) [1 pt] Consider `a=[]`. Which of the following statements will produce a different result from the others?

A. `a.append([1,2,3])`
B. `a.extend([1,2,3])`
C. `a = a + [1,2,3]`
D. `All of the above produce different results`
E. `None of the above produce different results`

(b) [1 pt] Which of the following is not a valid operator?
A. `%` B. `/` C. `//` D. `**` **E. All of these are valid operators**

(c) [1 pt] Which feature do Tuples and Strings have in common?

A. They are immutable

B. They are sequences

C. They can be used as dictionary keys

D. All of the above

E. None of the above

(d) [1 pt] Which of the following data types is iterable?

A. `str` B. `dict` C. `bool` D. `int` E. `float` **F. A and B** G. A, B, and D H. All of these

(e) [1 pt] A list can have a list as an element.

A. True B. False

(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

Use the following code to answer the next two questions.

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

(g) [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]`

(h) [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

(i) [1 pt] Which of the following statements are correct?

A. It is not possible to iterate through a dictionary.

B. `aDict.sort()` **will throw an exception.**

C. `aDict[ [1,2,3] ] = 'blue'` is a valid way to update a dictionary.

D. `aDict.get( 45, None)` will never return None.

E. A and B are valid.

F. C and D are valid.

(j) [1 pt] What would the `a` variable point at after the following code is executed?

```
try:
  a = 2
  if a % 2 == 0:
     a = a+3
  a = a / 0
  a = a * 2
except:
  a = a + 3
  a = a / 2
```

A. 6.5 **B. 4.0** C. 3.5 D. 3.0 E. 2.5 F. 2.0 G. 1.5

(k) [1 pt] Read the following segments of code.

```
a = 10
b = 20
c = 30
if a == 10:
   if b > 20:
      print ("first")
elif b == 20:
   print("second")
if c > 35:
   print("third")
elif c <=35:
   if a < 50:
      print("fourth")
   if b > 0:
      print("fifth")
   else:
      print("sixth")
else:
   print("seventh")
```

Select a piece of text that is printed when it is executed:

A. `first` is printed
B. `second` is printed
C. `third` is printed
D. `fourth` is printed
E. `sixth` is printed
F. `seventh` is printed

3. *(6 points)*
Fill in the blanks so that, when run, the code below will output the followng:

```
>>> func1()
DooWaa
Diddy
Diddy
Dum
Diddy
Doo
```

```
def func1():
    print( "DooWaa" )
    for i in range(  ____________________ ):

        print( ___________________)

        if i == _________________ :
            print("Dum")
    print("Doo")
```

Solution:

```
def func1():
    print("DooWaa")
    for i in range(___3___):
        print( __"Diddy"__)
        if i == ___1___ :
            print("Dum")
    print("Doo")
```

Grading: 2 points for each correct blank. -1 for any minor syntax errors. (leaving out quotes, etc)

4. *(4 points)*
Use the following code to answer the next two questions:

```
aList = [5,4,3,2,1]
bList = aList[1:4]
cList = []
cList.append(aList)
cList.append(aList)
```

(a) [2 pts] What is the contents of bList?

A. [4,3,2]

B. [5,4,3,2]

C. [4,3,2,1]

D. [5,4,3]

(b) [2 pts] Which of the following statements is true about cList?

A. cList is a copy of aList

B. cList is an alias of aList

C. cList[0] is a copy of aList

D. cList[0] is an alias of cList[1]

5. *(6 points)*

Pretend you are the Python interpreter and the following code has been entered and executed:

```
myList = [6, 3, (True, False), "Original"]
one = myList[:]
two = myList
three = one + ["Hello"]
one[2] = one[0] + one[1]
myList[1] = three[1:4:2]
two[3] = two[3]*2
one = one[2:]
three[0] = one[0]
```

For each of the three new lists (one, two, three), which are aliases and which are copies?

What are the values of each of the following variables:

myList =

one =

two =

three =

Solution: aliases = two
copies = one, three

myList = [6, [3, 'Original'], (True, False), 'OriginalOriginal']
one = [9, 'Original']
two = [6, [3, 'Original'], (True, False), 'OriginalOriginal']
three = [9, 3, (True, False), 'Original', 'Hello']

Grading: +2 points for correctly identifying each alias/copy.
+1 point for each correct list.
-0.5 points if they use a tuple instead of a list.

6. *(8 points)*

Write a function named `countCapitals` that takes in a string as its only parameter, and returns an integer count of the number of capital letters in the string.

Example run:

```
>>> result = countCapitals( "Hello There Wonderfull CS Student" )
>>> print( result )
6
>>>
```

Solution:

```
import string

def countCapitals( aString ):
   count = 0
   for item in aString:
      if item in string.ascii_uppercase:
         count = count + 1

   return count
```

Grading:
+1: Correct function header
+2: Itterates through the contents of the string
+2 check for capital letters correctly.
+2: correctly increments counter.
+1: returns the integer result.

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)*

Write a function named `rankDays` which takes in one parameter, aList. The list will be a list of tuples, with each tuple containing information in the format (Number Processed, Day of Week). The Day of Week will be a string (either "Monday", "Tuesday", "Wednesday", "Thursday", or "Friday"), and the Number Processed will be an integer. You should create and return a dictionary which has the day's "rank" (a number 1-5, with 1 corresponding to the day with the largest number of items processed) as a key and the corresponding day of the week as the value. The ranking criteria will be on the numbers of items processed each day. You may assume that no two days will share the same number of items processed (i.e. each day will have a uniquely different number of items processed).

Example test case:

```
>>> aList = [(456, "Monday"), (371, "Tuesday"), (2316, "Wednesday"),
(1371, "Thursday"), (17, "Friday")]
>>> myDict = rankDays(aList)
>>> print(myDict)
{1: 'Wednesday', 2: 'Thursday', 3: 'Monday', 4: 'Tuesday', 5: 'Friday'}
```

Solution:

```
def rankDays(aList):
   myDict = {}
   aList.sort(reverse = True)
   rank = 1
   for item in aList:
      myDict[rank] = item[1]
      rank = rank+1
   return myDict
```

Grading: +1 correct function header
+1 uses dictionary to store values
+2 the keys in the dictionary are the numbers 1..5
+3 correctly ranks days based upon number of items processed.
(-1 if sort order is backwards)
+2 the value of each key is the correct day in the week.
+1 returns a dictionary
-2 if assigning none to a critical variable (e.g. alist=alist.sort())