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 7 questions on 8 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. Multiple Choice	7	
3. Find The Errors	8	
4. Indexing	5	
5. List Contents	4	
6. sumFloats	8	
7. combineNames	10	
Total:	51	

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] boolean expression

Solution: A python statement that evaluates to either True or False. Typically contains a comparison operator (<, >, <=, >=, ==, !=) and may contain boolean operators such as AND, OR, or NOT.

(b) [3 pts] evaluate

Solution: To simplify an expression by performing the operations in order to yield a single value.

(c) [3 pts] slice

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

2. *(7 points)*
For each of the following multiple choice questions, indicate the single most correct 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] 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.

(f) [1 pt] Examine the following code:

```
aDict = {'GT Baseball': ['8 wins', '5 losses'],
 'GT Mascot Challenge': '13-0', 'GT Basketball': '11-5',
 'GT Football': '8-4'}
keys = aDict.keys()
values = aDict.values()
```

Using the `keys` variable, how does one put every key from the dictionary, aDict, into a list?

A. Use the has_key() method to move keys into a list.

B. Iterate through keys and append each item to an originally empty list.

C. Set a variable equal to `keys.list()`

D. Set a variable equal to `keys.getList()`.

(g) [1 pt] (continued...) The following line of code is run AFTER the three lines of code above have already ran. What does does it do?

```
aDict['GT Water Polo'] = ['10 wins', '1 loss', '4 ties']
```

A. Adds the string, 'GT Water Polo', as a value and the list `['10 wins', '1 loss', '4 ties']`, as a key to aDict.

B. Overwrites the old aDict and sets aDict equal to `{'GT Water Polo': ['10 wins', '1 loss', '4 ties']}`.

C. Adds the string, 'GT Water Polo', as a key and the string, '10-1-4', as a value to aDict.

D. Adds the string, 'GT Water Polo', as a key and the list `['10 wins', '1 loss', '4 ties']`, as a value to aDict.

3. *(8 points)*

The following function is supposed to take in a file name as a parameter, open the file, and print out every line of the file one at a time.

However, there are four errors in the code. Indicate what each error is, and tell us what code you would change or add to fix each error.

```
def fileReader(fileName):

    f = open(fileName, 'w')

    line = f.readline()

    while line != '':
        line = f.readline()
        print(f, end='')

    f.close
```

Solution: Two points for each error. One point for identifying the error, and one point for correctly fixing it.

1. The open statement should be in read mode ("r" vs "w").
2. The while loop should print the line before getting the next line! As it is, it currently skips printing the first line!
3. The print statement should say `print(line, end='')`
4. The last line needs parenthesis to call the close method

4. *(5 points)*

Execute the following piece of code as if you were the python interpreter.

```
aList = [13,4,5,10.4,9,0]
bList = []
for pos in range(4):
    if aList[pos]//3 == 3:
```

```
      bList.append(aList[pos])
   if aList[pos]%3 < 2:
      aList.remove(aList[pos])
   elif type(aList[pos]) != type(0.0):
      bList = bList +  [ aList[pos] ]
```

Write the contents of aList and bList after the above code is executed.

aList -

bList -

Solution:

```
 aList => [4, 5, 9]
 bList => [5, 10.4]
Grading:
 +1 point for each correct number in the correct list.
 -1 for each extra number in either list.
```

5. *(4 points)*
Write what is printed to the screen after the following code is executed:

```
a = ['a', 'b', 'c']
b = "2316"
c = [a, b, 5]
d = c[:]

for x in d[1]:
    a.append(x)
    c[2] = c[2] + 1

print(a)
print(b)
print(c)
print(d)
```

Solution:

```
['a', 'b', 'c', '2', '3', '1', '6']
2316
[['a', 'b', 'c', '2', '3', '1', '6'], '2316', 9]
[['a', 'b', 'c', '2', '3', '1', '6'], '2316', 5]

+1 for first list exactly correct
+1 for 2316 (not in a list or quotes)
+1 for 3rd line/list exactly correct
+1 for 4th line/list exactly correct
```

6. *(8 points)*
Write a function named `sumFloats` that takes in list of data, and adds together all of the floats in the list, returning the resulting value.

Your function should go through each item in the original list and check to see if it is a float. If it is a floating point number, you should add it to a running total. When you are finished, return the running total. Do not add integers or other types!

Example run:

```
>>> result = sumFloats( [10.0, 5, True, 'Testing', 11.2] )
>>> print( result )
21.2
>>>
```

Solution:

```
def sumFloats( aList ):
   total = 0.0
   for item in aList:
      if type(item) == float:
         total = total + item

   return  total
```

Grading:
+1: Correct function header
+1: Iterates through the contents of the list
+2 check for float type correctly.
+2: correctly adds item to the total.
+1: Does not add non-floats.
+1: returns the total.

7. *(10 points)*

Write a function named `combineNames` that takes in two lists of strings. The first list will have first names, and the 2nd list will have matching last names. You may assume that both lists are of the same length.

Your function should go combine first names with last names, and return a new list with a "lastname, firstname" string for each pair. The list you return should be sorted by last name!

Example run:

```
>>> result = combineNames( ['Jay', 'Jake', 'Sam'], ['Summet', 'Evans', 'Kass'] )
>>> print( result )
[ 'Evans, Jake', 'Kass, Sam', 'Summet, Jay']
>>>
```

Solution:

```
def combineNames(fnList, lnList ):
   allNames = []
   for index in range( len(fnList) ):
```

```
      fn = fnList[index]
      ln = lnList[index]
      name = ln+", "+fn
      allNames.append(name)
   allNames.sort()
   return allNames
```

Grading:
+1: Correct function header
+4: Iterates through the contents of both lists (matching them)
+2: creates "lastname, firstname" string correctly (w/ space and comma)
+2: Sorts resulting list +1: returns the result.