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 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	9	
2. Multiple Choice	7	
3. For/If/List	3	
4. List Operations	7	
5. DooWaa	6	
6. Count Capitals	8	
7. trainCrash	5	
8. addItems	10	
Total:	55	

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] nested loop

   Solution: A loop inside the body of another loop.

   (c) [3 pts] syntax error

   Solution: An error in a program that makes it impossible to parse – and therefore impossible to interpret.

2. *(7 points)*
For each of the following multiple choice questions, indicate the best correct answer. Indicate your selected answer by circling it.

   (a) [1 pt] The list `[2,1,4,3,"5"]` can not be sorted by .sort() because the last element is a string. **A. True** B. False

   (b) [1 pt] The list `[2,1,4,3,"5"]` can not be reversed by .reverse() because .reverse() will fail when there is a string in a list of numbers.

   A. True **B. False**

(c) [1 pt] Which name below will give an error message because it is **not** a valid Python variable name?

A. `picture-1`
B. `picture_1`
C. `picture1`
D. `picture`
E. None of the above

(d) [1 pt] Assume the following code has been executed by the python interpreter:

```
def mysteryFunc():
    x = 17
    return print(x)

def mysteryFunc2():
    for x in range(5,15):
        print(x)
        return x

x = [1,2,3]
x = x.append(4)
a = mysteryFunc()
b = mysteryFunc2()
```

Which of the following is correct?

A. The value in x is `[1, 2, 3, 4]`
B. The value in a is `17`
C. the value in b is `14`
D. the value in b is `15`
E. The value in b is `5`
F. The value in b is `None`

(e) [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

(f) [1 pt] Which of these statements is a legal expression?

A. "A"+ "B"
B. "A" - "B"
C. "A" * "B"
D. "A" / "B"

(g) [1 pt] Assume the following code has been executed by the python interpreter:

```
dataType = 2**4*1/(3%2)
```

What is the type of the data referenced by the dataType variable?

A. String B. Integer **C. Float** D. List E. NoneType

3. *(3 points)*
Consider the following lines of code:

```
aList = [1, 2, 3, 4, 5, 6]
bList = []
cList = []
for item in aList:
        if item % 2 == 0:
                bList.append(item)
                out = aList[-1]
                del aList[-1]
                cList = cList + [out]
```

What is stored in each variable after the code is executed?

aList =

bList =

cList =

Solution: aList = [1, 2, 3, 4]
bList = [2, 4]
cList = [6, 5]

Grading: +1 for each correct answer.
(-1 overall if they forget square brackets.)

4. *(7 points)*
Read the following pieces of code. After the python interpreter executes the code, what is stored in the cList, bList, and aList variables?

```
>>> aList = [7,9,12]
>>> bList = aList[:]
>>> cList = aList
>>> cList[2] = 'cheese'
>>> aList = ['pizza', [5, '6', bList], aList]
```

(a) [3 pts] aList is:

Solution: ['pizza', [5, '6', [7, 9, 12]], [7, 9, 'cheese']]
grading: 1 point for ['pizza', [5,'6',[7,9,12]]
2 points for the [7,9,'cheese']

(b) [2 pts] bList is:

Solution: [7, 9, 12]

(c) [2 pts] cList is:

Solution: [7, 9, 'cheese']

5. *(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)

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:

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

```
def trainCrash (x):
   while x <= 10:
       if x % 5 == 0:
           return "oh no, Crash!"
       if x % 3 == 0:
           print("I'm a train...")
       x = x +1
       print("Choo-choo!")
   return "I'm too tired to go on"
```

If this code is called from the IDLE window as follows:

```
y = trainCrash(8)
```

(a) [3 pts] What is displayed on the screen?

Solution:

```
Choo-choo!
I'm a train...
Choo-choo!
```

Grading: 1 point for each correct line, -1 point for each extra line

(b) [2 pts] What will be stored in the y variable from the example function call above?

Solution:

```
'oh no, Crash!'
```

Grading: 2 points for the string stored in y.

8. *(10 points)*

Write a function called `addItems` that accepts two lists as parameters. It will return a list where each element is the sum of the elements at that position in the two lists given to the function. You may assume that the two lists will be of the same length, and that the items in each position are valid to add together using the + operator.

For example:

```
>>> result = addItems( [1,2,3, 'hi'] , [1,1,1, ' there'] )
>>> print( result )
 [2, 3, 4, 'hi there']
```

Solution:

```
def addItems(aList, bList):
  cList = []
  for index in range( len(aList) ):
    added = aList[index] + bList[index]
    cList.append(added)
  return cList
```

Grading:
1 pt - Correct def statement
1 pt - Itterates through aList
1 pt - itterates through bList
3pt - correct matches items from aList to items in bList
2pt - stores the summed result correctly.
1pt - returns a list.
1pt - returns correct list.